楽しく学べる

Quick BASIC

松尾昭明



東京書籍

楽しく学べる
Quick BASIC

松尾昭明

東京書籍

はじめに

最近のパソコンの普及は著しく、企業をはじめ学校や家庭でもパソコンを利用するようになりました。その多くはワープロや表計算のソフトを使っていますが、中にはＢＡＳＩＣで仕事に合ったプログラムを作成したり楽しんだりしようとしている方もおられることでしょう。

一般にＢＡＳＩＣの入門書で独習される場合が多いのですが、せっかく意欲をもって始めたものの、途中で挫折される方も少なくありません。それは、入門書の多くが文法を中心に説明していることから、個々の文法は理解できても実用的なプログラムの技法を習得できないことや、無味乾燥な文法の説明に興味関心を失ってしまうところに、原因があると思われます。

そこで本書は、ＢＡＳＩＣの文法にこだわらずグラフィックスから導入することによって、プログラミングの楽しさを味わいながら次第に実際的なプログラミング技術を身につけるよう構成しました。

本書が利用したＱｕｉｃｋ　ＢＡＳＩＣは、構造化プログラミングができる言語ですから他のＢＡＳＩＣよりも使いやすく、わかりやすいプログラミングが可能です。表計算ソフトやデータベースソフトの利用とともに、Ｑｕｉｃｋ　ＢＡＳＩＣのプログラミングによってパソコンがもっと楽しいものになってほしいと願っております。

また、本書の編集に当たって実際に高校生の実習テキストとして利用したところ、特別に説明をしないで本書のみでプログラミングを習得させることができました。このことからも、本書を高校生の実習テキストとしてお奨めするとともに、本書によってＱｕｉｃｋ　ＢＡＳＩＣの実用的なプログラミングを楽しみながら独習できるものと期待いたしております。

なお、プログラムの作成に当って使用した主なハード及びソフトは次のとおりです。

パソコン本体	PC9801VM2,PC9801VX2,PC9801NS20
プリンタ	PCPR201G,M1724P
ＣＲＴ	PCKD551,PCKD854
ハードディスク	CRCMH4B,CRCMH2B
モデム	ｵﾑﾛﾝMD2400B,ｵﾑﾛﾝMD24FN5,COMSTARZ2424/5
ソフト	Quick BASIC Ver 4.5

松尾昭明

目次

第1章

準備

購入したQuick BASICを実際に使えるようにします。

１．１　チョット我慢して！！

Ｑｕｉｃｋ　ＢＡＳＩＣを購入されたあなたは、すぐにはじめたい気持ちで胸がいっぱいでしょう。でもチョット待ってください。

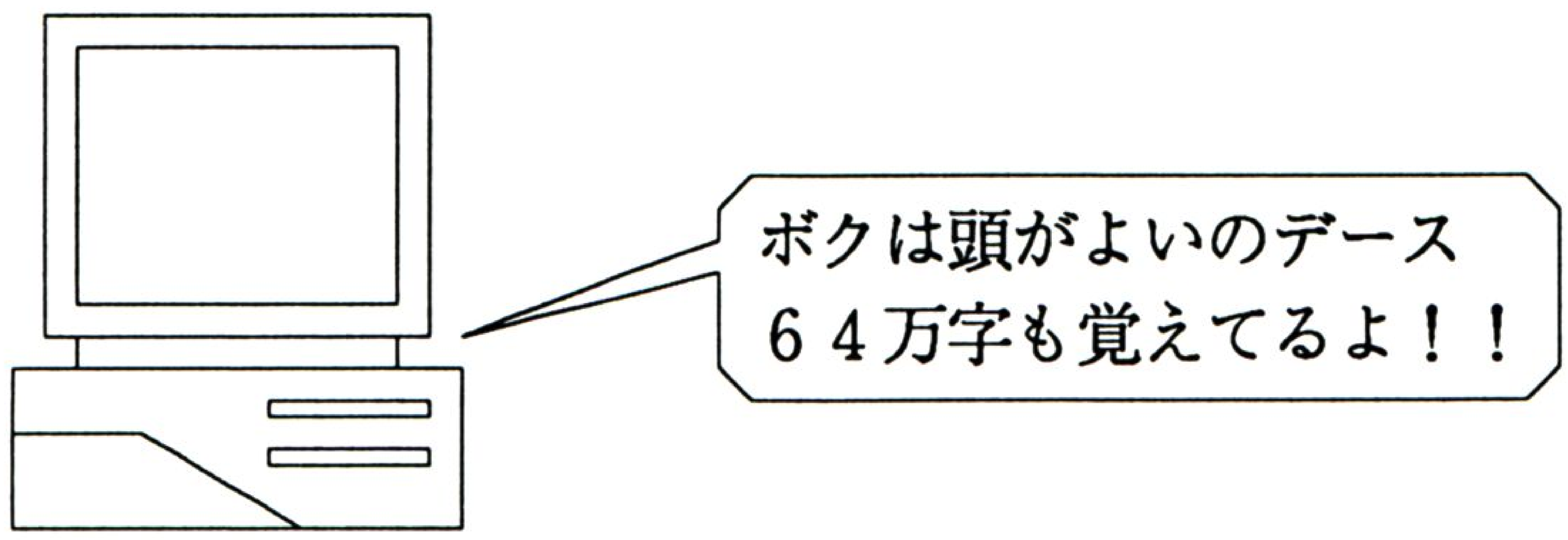
はじめに、Ｑｕｉｃｋ　ＢＡＳＩＣを利用するためのハードウェアやソフトウェアを確認しましょう。

1. パソコンに適合したソフトを購入してください。

パソコンはＰＣ９８００シリーズ、ＦＭＲ５０シリーズ、ＰＡＮＡＣＯＭＭ５００シリーズなどです。

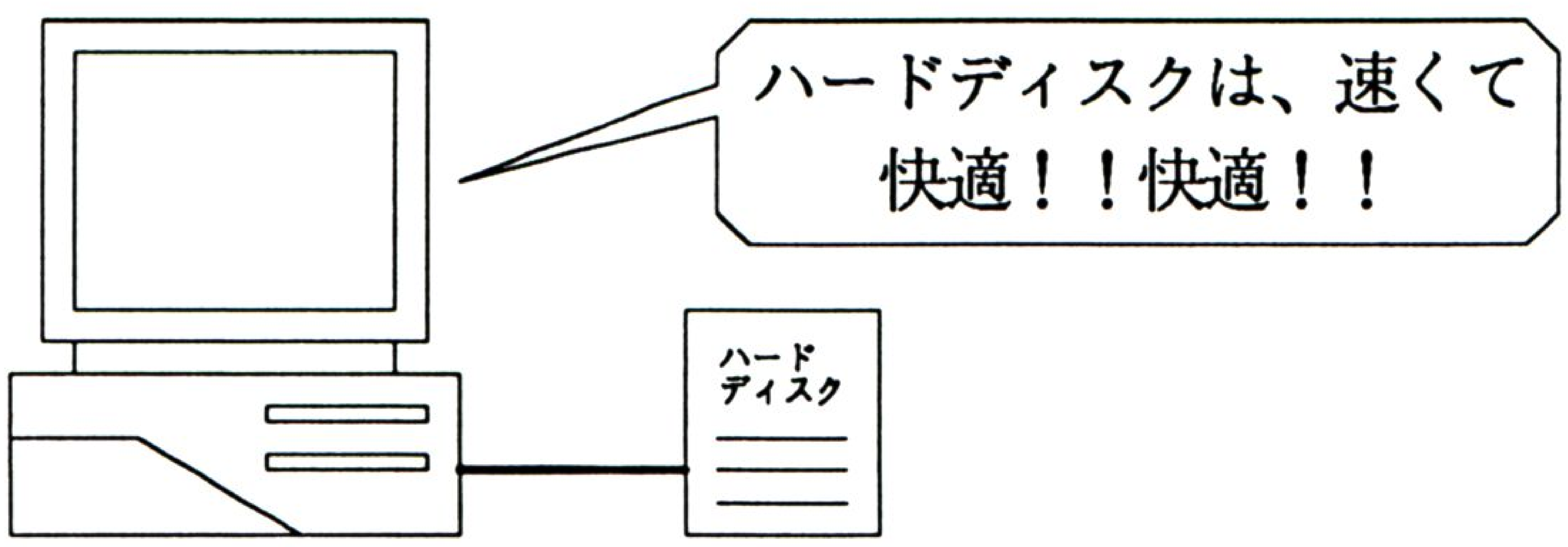
パソコンの記憶容量は、６４０ＫＢ（キロバイト）以上を必要とします。最近の機種は６４０ＫＢが標準装備されていますが、以前の機種の中には３８４ＫＢのものもあります。その場合は増設して６４０ＫＢにしてください。

６４０ＫＢというのは、英数字で６４万字相当を記憶できるということです。Ｑｕｉｃｋ　ＢＡＳＩＣはプログラムが大きいので、記憶容量が少ないと仕事ができないのです。



2. ディスク装置は、パソコンに付いている２ドライブのフロッピーディスク装置で十分です。

ただし、Ｑｕｉｃｋ　ＢＡＳＩＣは多くのプログラムやデータから構成されているので、読み書きの速いハードディスクのほうが快適に利用できます。

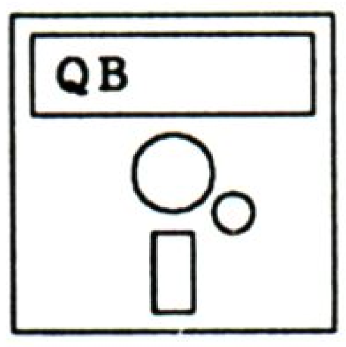


3. ＭＳ－ＤＯＳのフロッピーディスクを用意します。

ＭＳ－ＤＯＳは、マイクロソフト社が開発して発売しているオペレーティングシステムです。今のところ、世界のパソコンの多くがこのソフトを基本ソフトとして利用しています。

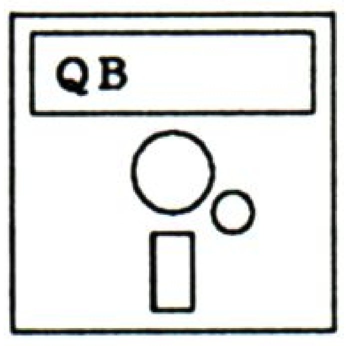
４. もちろん、Ｑｕｉｃｋ　ＢＡＳＩＣ　Ｖｅｒ４．５を用意します。

Ｑｕｉｃｋ　ＢＡＳＩＣの箱を開けると、以下のフロッピーディスクが入っています。



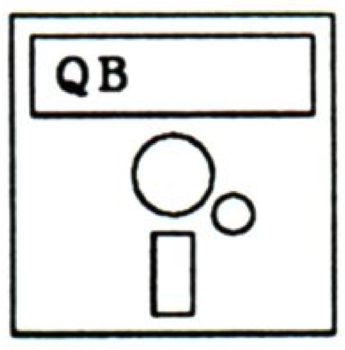
・セットアップディスク

Ｑｕｉｃｋ　ＢＡＳＩＣが快適に使えるよう準備するためのプログラムが入っています。



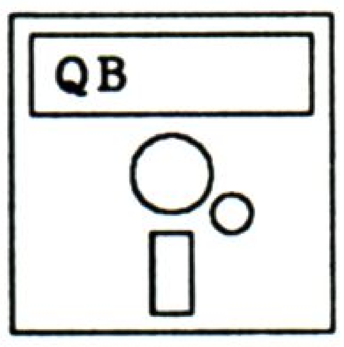
・プログラムディスク

Ｑｕｉｃｋ　ＢＡＳＩＣの最も重要なプログラムが入っています。



・ライブラリーディスク

Ｑｕｉｃｋ　ＢＡＳＩＣのプログラムの中で利用できる便利なプログラムのライブラリが入っています。



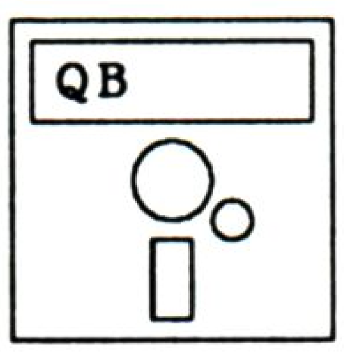
・アドバイザディスク

Ｑｕｉｃｋ　ＢＡＳＩＣの使い方や解説が入っています。

そして、厚い厚いマニュアルが入っています。マニュアルは、Ｑｕｉｃｋ　ＢＡＳＩＣの文法等について詳細に解説していますが、カタカナの専門用語が多数使われているので読みこなすには相当の努力を必要とします。１ページ目から順に読んで理解しようとするのではなく、本書でプログラミングに馴れながら、辞書を引くように利用してください。

５. 最後に、漢字の変換システムを用意します。

Ｑｕｉｃｋ　ＢＡＳＩＣで漢字を利用するためには、次のような漢字の変換システムを必要とします。



ＰＣ９８００シリーズ	ＭＳ－ＫＡＮＪＩ	ＡＴＯＫ６
ＦＭＲ５０シリーズ	ＭＳ－ＫＡＮＪＩ	ＯＡＫ

ＡＴＯＫ６は「一太郎Ｖｅｒ３．０」のシステムディスクに入っています。一太郎には「Ｖｅｒ４．３」や「一太郎ｄａｓｈ」もあるのですが、これらに付属しているＡＴＯＫ７は利用できないようです。

ＯＡＫはＦＭＲ５０のＭＳ－ＤＯＳシステムディスクに入っています。

１．２　さあ！準備です！！

いよいよ、Ｑｕｉｃｋ　ＢＡＳＩＣを実際に使えるように準備を始めます。
この準備のことを「インストール」といいます。パソコンの入門書やマニュアルには必ず出てくる言葉なのですが、この用語のために準備の段階でつまずいたりするものです。「インストール」という言葉にはこだわらないで、次に進みましょう。

これからの仕事は、次の２つです。

１．新しいフロッピーディスクをフォーマットします。

２．Ｑｕｉｃｋ　ＢＡＳＩＣのディスクからフォーマットしたフロッピーディスクに必要なファイルを複写して実行用のフロッピーディスクを作ります。

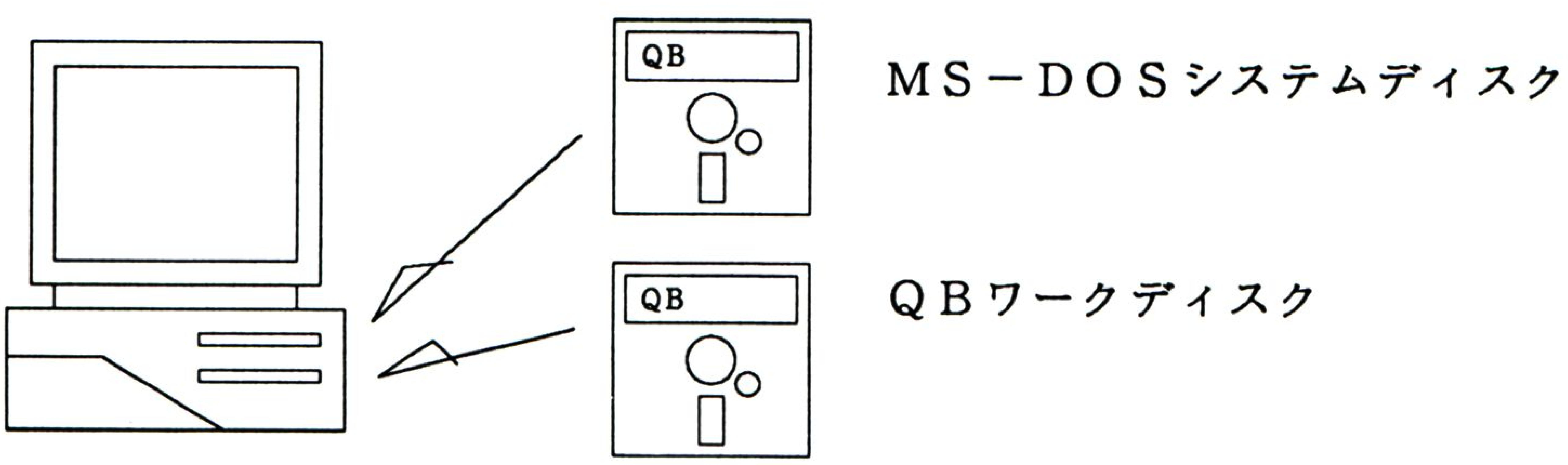
1. 新しいフロッピーディスクをフォーマットします。

フォーマットするということは、広い原野を区切って団地を造成して、何丁目何番地というように番地を付けるようなものです。原野に無秩序に家を建てたのでは、どこに誰が住んでいるのかわからなくなるではありませんか。フロッピーディスクも、素早くデータを読み書きするために記憶場所に番地を付けるのです。

1.電源をＯＮにして、Ａドライブ＾ＭＳ－ＤＯＳシステムディスクをセットし、リセットボタンを押してＭＳ－ＤＯＳを起動してください。

2.新しいフロッピーディスクを３枚用意して、「ＱＢワークディスク」、「ＱＢアドバイザディスク」、「ＱＢ起動ディスク」のラベルを貼っておきます。

3.「ＱＢワークディスク」をＢドライブにセットしてから、**FORMAT B:** と入力してフォーマットします。以下は画面の指示に従って進めてください。




```
A>FORMAT B:
```

入力後は ↵ を押す。（以下同じ）

4.「ＱＢワークディスク」のフォーマットが終ったら、「ＱＢアドバイザディスク」をＢドライブにセットし、同様にフォーマットをします。

5.次に「ＱＢ起動ディスク」をＢドライブにセットしたら、今度はFORMAT B: /Sと入力して、システムフォーマットをします。

/Sは、ＭＳ－ＤＯＳのシステムのうち IO.SYS、MS-DOS.SYS及びCOMMAND.COMをフロッピーディスクの先頭部分に複写することです。IO.SYS及びMS-DOS.SYSは隠しファイルなので DIRコマンドでは見ることができません。

```
A>FORMAT B: /S
```

2. 購入したＱｕｉｃｋ　ＢＡＳＩＣのフロッピーディスクから、実行用のフロッピーディスクを作成します。

ディスクの差し替えが面倒ですが、間違ったら CTRL （コントロールキー）と Ｃ とを同時に押すと、作業を中断できます。

1.セットアップディスクをＡドライブへセットして、SETUPを入力します。

```
A>SETUP
```

2.たくさんの説明が表示されますから、これを読みながら操作をすればよいのです。わかりにくい専門用語はホドホドにして「何かキーを押してください。」というメッセージに従ってキーを押していくと、次の画面が表示されます。メニューの選択は、すでに１が表示されていますから ↵ を押してください。

```
ＱｕｉｃｋＢＡＳＩＣ　セットアップ　ユーティリティ

***********＜メニュー＞******************************
　１．　ＱｕｉｃｋＢＡＳＩＣの利用環境を作成します。
　２．　ＱＢの概要、操作方法の説明を行います。
　３．　セットアップを終了します。
*****************************************************

●メニューを選択してください。・・・・・・・・・・？１
```

3.すると、次から次へと入力を求められますが、いずれも画面にあらかじめ表示されているのでそのまま ↵ を押してください。

```
ＱｕｉｃｋＢＡＳＩＣ　セットアップ　ユーティリティ

　　●転送先のドライブ名を設定してください。？Ａ
　　●どのドライブへインストールしますか？Ｂ

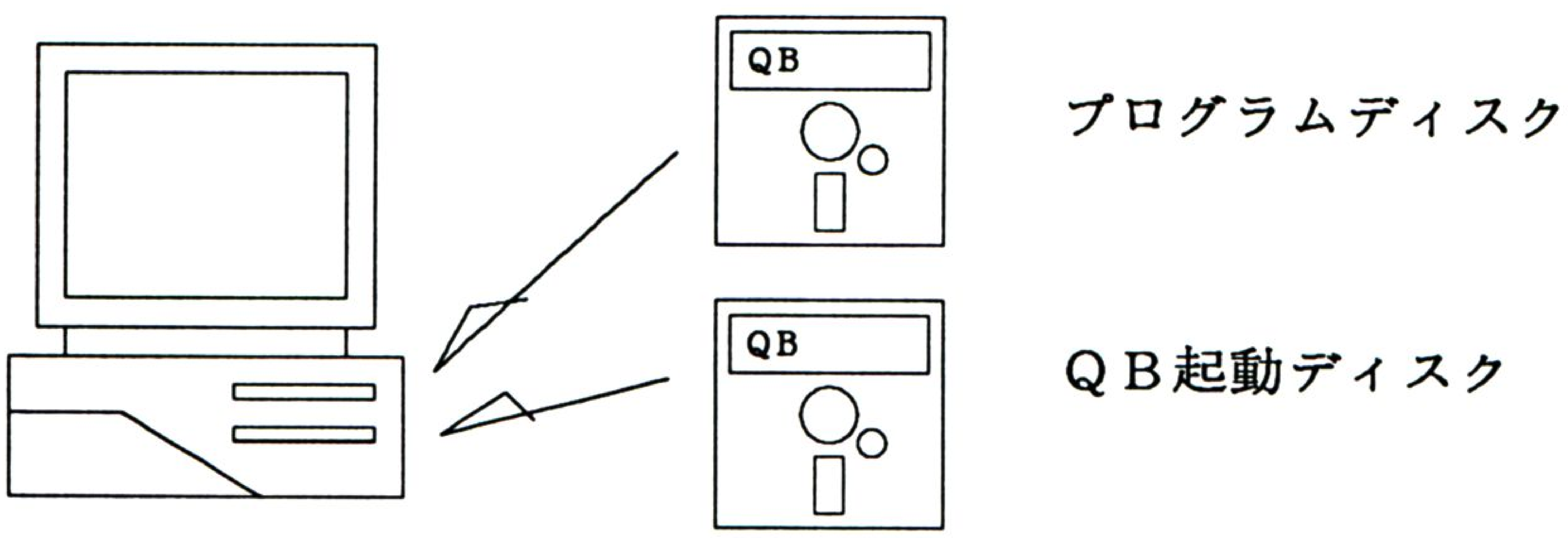
　　●上記の設定でよろしいですか[N/Y]？Ｙ
```

```
ＱｕｉｃｋＢＡＳＩＣ　セットアップ　ユーティリティ

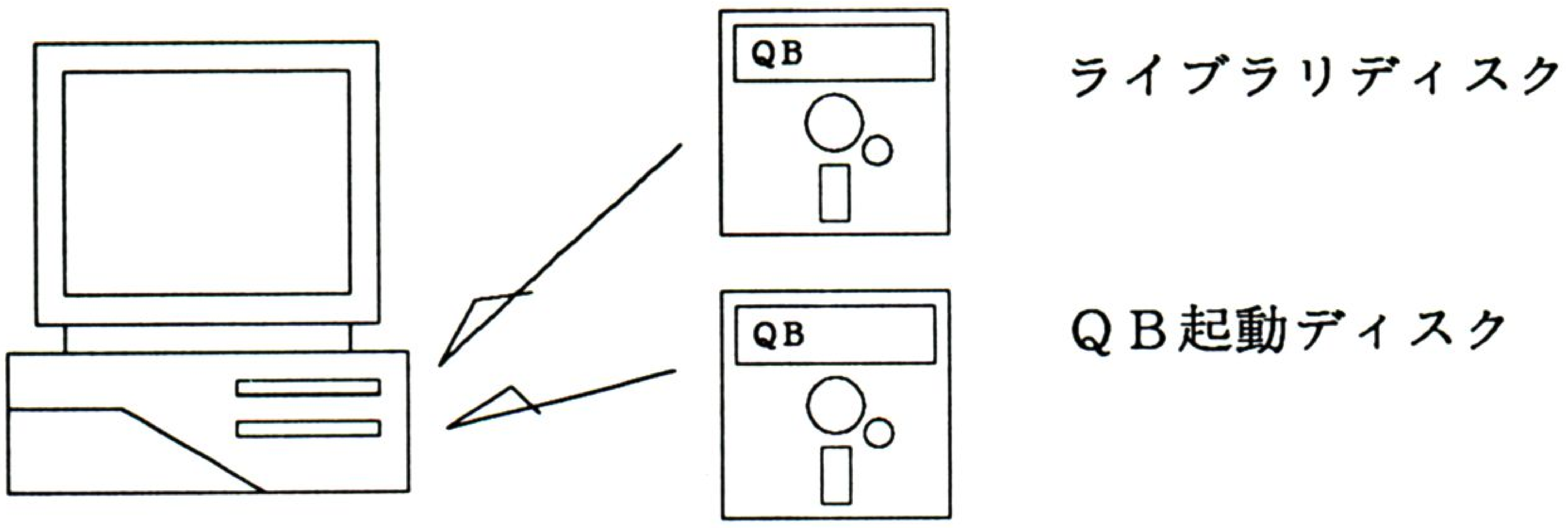
●数値演算ライブラリを選択してください[1:Alt Math / 2 Emu･･･]？1
●マウスを使用しますか[Y/N]？Ｙ
●マウスの種類を選択してください[1:バスマウス / 2:シリ････]？1
●マウスドライバを選択してください[1:MOUSE.SYS / 2:MOUSE.COM]？2
●ドキュメントファイルを転送しますか[Y/N]？Ｙ
●サンプルプログラムを転送しますか[Y/N]？Ｙ

●上記の設定でよろしいですか[N/Y]？Ｙ
```

4.「プログラムディスク」をＡドライブに、さきほどフォーマットした「ＱＢ起動ディスク」をＢドライブにセットして、何かキーを押します。
　すると、Ｑｕｉｃｋ　ＢＡＳＩＣの「プログラムディスク」から必要なファイルが「ＱＢ起動ディスク」へ複写されます。

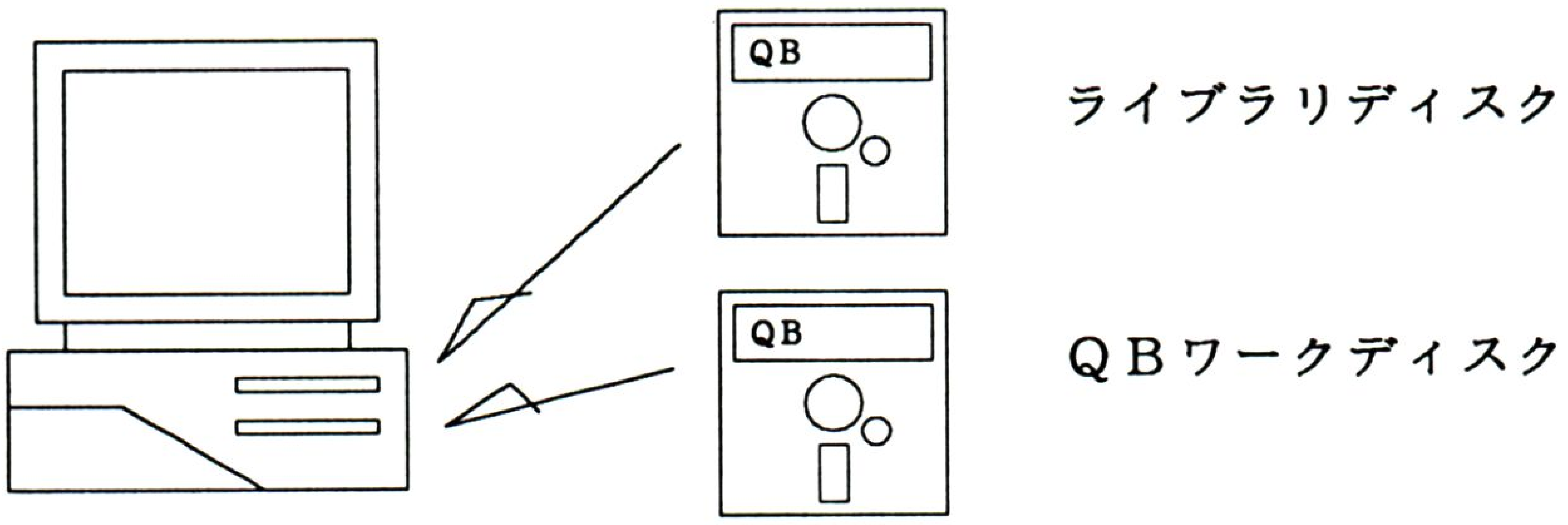


5.Ａドライブに「ライブラリディスク」をセットして、何かキーを押します。
　すると、Ｑｕｉｃｋ　ＢＡＳＩＣの「ライブラリディスク」から必要なファイルが「ＱＢ起動ディスク」へ複写されます。

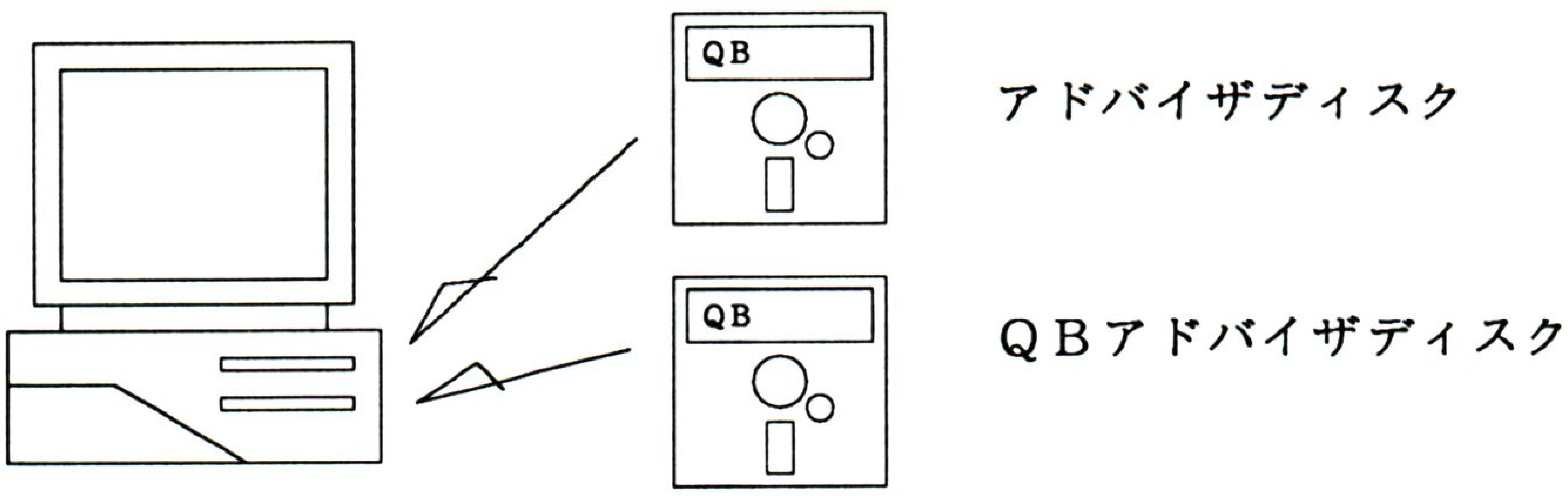


6.Ｂドライブにさきほどフォーマットした「ＱＢワークディスク」をセットして、何かキーを押します。

すると、Ｑｕｉｃｋ　ＢＡＳＩＣの「ライブラリディスク」から必要なファイルが「ＱＢワークディスク」へ複写されます。

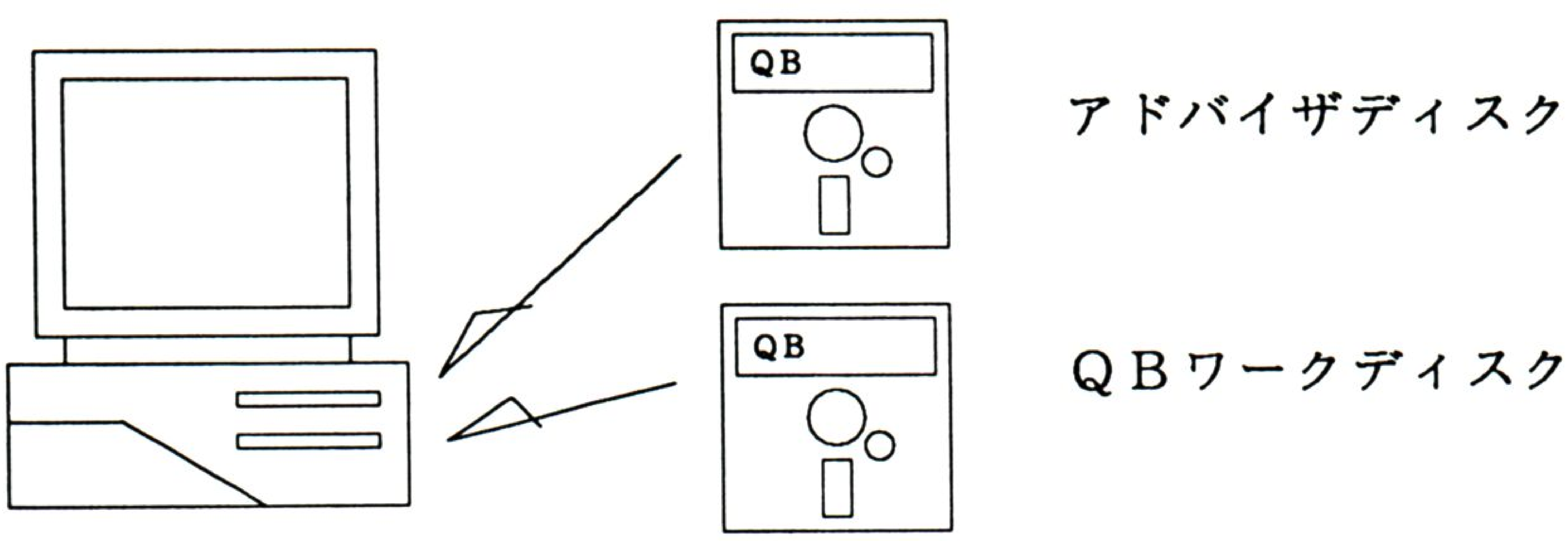


7.Ａドライブに「アドバイザディスク」をセットして何かキーを押し、続いてＢドライブにさきほどフォーマットした「ＱＢアドバイザディスク」をセットして、何かキーを押します。Ｑｕｉｃｋ　ＢＡＳＩＣの「アドバイザディスク」から、必要なファイルが「ＱＢアドバイザディスク」へ複写されます。

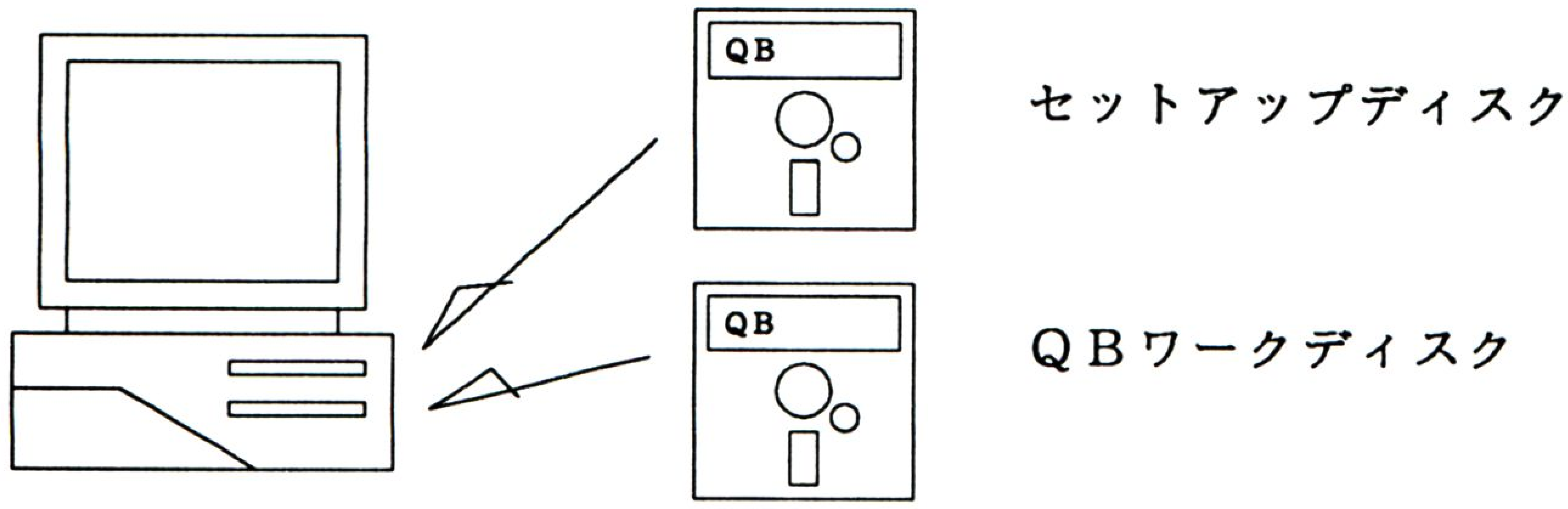


8.Ｂドライブに「ＱＢワークディスク」をセットして、何かキーを押します。

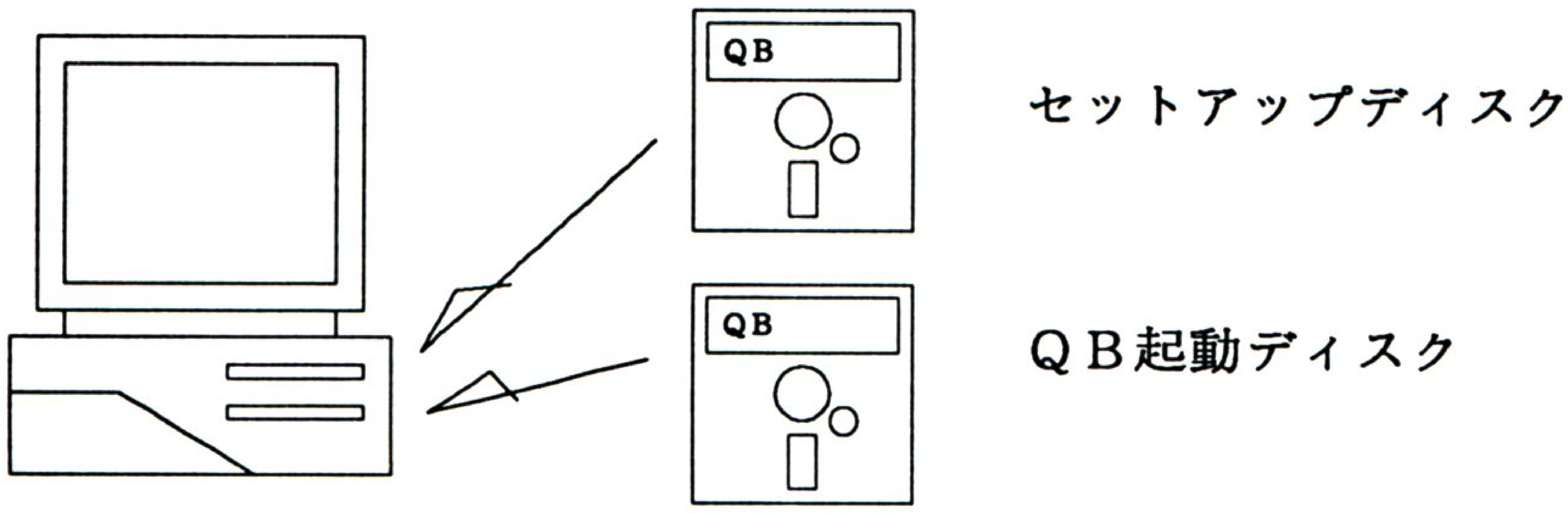
すると、Ｑｕｉｃｋ　ＢＡＳＩＣの「アドバイザディスク」から必要なファイルが「ＱＢワークディスク」へ複写されます。



9.Ａドライブに「セットアップディスク」をセットして、何かキーを押します。
　すると、Ｑｕｉｃｋ　ＢＡＳＩＣの「セットアップディスク」から、必要なファイルが「ＱＢワークディスク」へ複写されます。



10.利用環境を作成します。
　何かキーを押してから、Ｂドライブに「ＱＢ起動ディスク」をセットして、再び何かキーを押します。



これで、Ｑｕｉｃｋ　ＢＡＳＩＣの実行用ディスクの作成は終了しました。

3. 次にメニューの２を選択して、Ｑｕｉｃｋ　ＢＡＳＩＣの概要と操作方法の説明を読んでください。

```
ＱｕｉｃｋＢＡＳＩＣ　セットアップ　ユーティリティ

***********＜メニュー＞*****************************
　１．　ＱｕｉｃｋＢＡＳＩＣの利用環境を作成します。
　２．　ＱＢの概要、操作方法の説明を行います。
　３．　セットアップを終了します。
****************************************************

●メニューを選択してください。・・・・・・・・・・？２
```

初めはＱｕｉｃｋ　ＢＡＳＩＣの雰囲気を味わう程度でよいでしょう。ここに

説明されているすべてを理解できなければ前に進めないというものではありません。初心者にとっては、多数の専門用語がわからなくて当然です。気にしないで「今にわかるさ。」という気持ちでご覧ください。

4. 新しく作成した実行用のフロッピーディスクの内容は次のとおりです。

1.ＱＢ起動ディスク

```
┌── IO.SYS   MSDOS.SYS   COMMAND.COM   AUTOEXEC.BAT   CONFIG.SYS
├── BIN ──── QB.EXE   QB.INI   BRUN45A.EXE   LINK.EXE   LIB.EXE   BC.EXE
│            MOUSE.COM
└── LIB ──── BQLB45.LIB   QB.LIB   QB.QLB   BCOM45A.LIB   BRUN45A.LIB
             GEN.LIB   GEN.QLB   GRAPH.LIB   GRAPH.QLB   MOUSE.LIB
             MOUSE.QLB
```

2.ＱＢワークディスク

```
┌── INCLUDE ─ GEN.BI   GRAPH.BI   MOUSE.BI   QB.BI   GNCONST.BI
│             WHCONST.BI
├── SOURCE ── DEMO1.BASなど
│             ADVR_EX              CALL_EX.BASなど
│             EXAMPLE              GENDEMO1.BASなど
│             BOOK                 CHAP210.BASなど
├── ADVR ──── QB45QCK.HLP   QB45ENER.HLP   QB45USER.HLP
└── DOC ───── README.DOC   SAMPLE.DOC
```

3.ＱＢアドバイザディスク

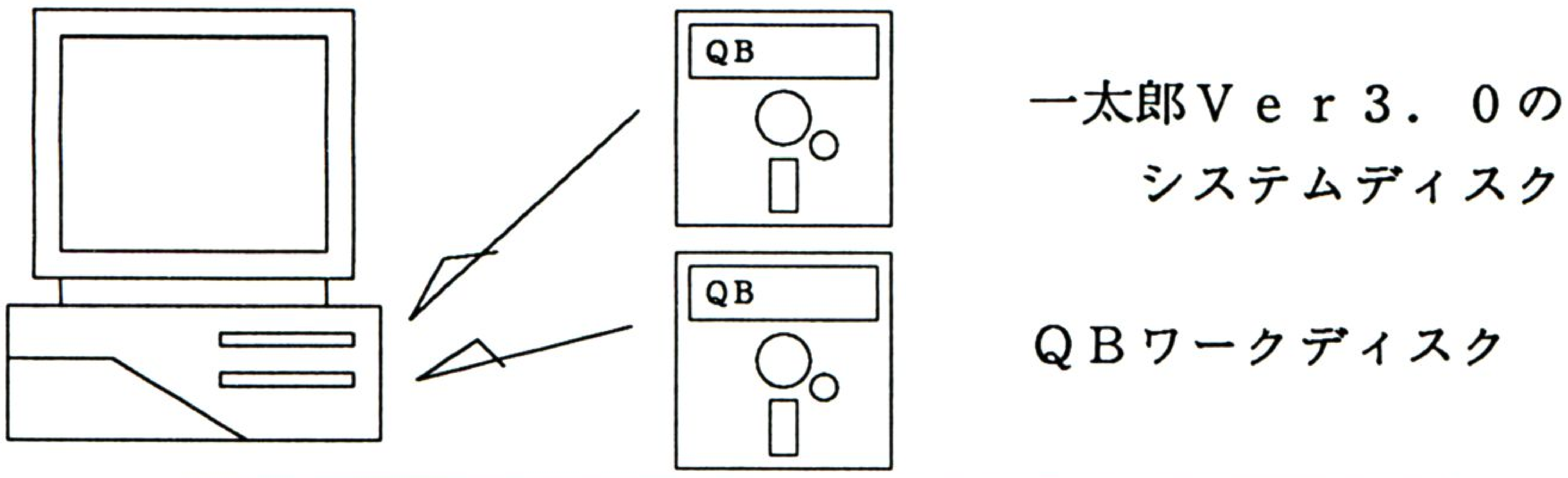
```
─── ADVR ──── QB45QCK.HLP   QB45ENER.HLP   QB45USER.HLP
              HELPMAKE.EXE
```

１．３　日本語を使いましょう

漢字やひらかなを使用するには、漢字の変換システム（日本語フロントエンドプロセッサ）を組み込む必要があります。

ＰＣ９８００シリーズ用のＱｕｉｃｋ　ＢＡＳＩＣが利用できるフロントエンドプロセッサは、ＭＳ－ＫＡＮＪＩ及びＡＴＯＫ６です。ここではワープロソフトの「一太郎Ｖｅｒ３．０」に組み込まれているＡＴＯＫ６を利用することにします。

1. 「一太郎」のシステムディスクをＡドライブに、「ＱＢワークディスク」をＢドライブにセットして、COPY ATOK*.* B: と入力してください。ＡＴＯＫ６に必要な辞書などのファイルが「ＱＢワークディスク」へ複写されます。

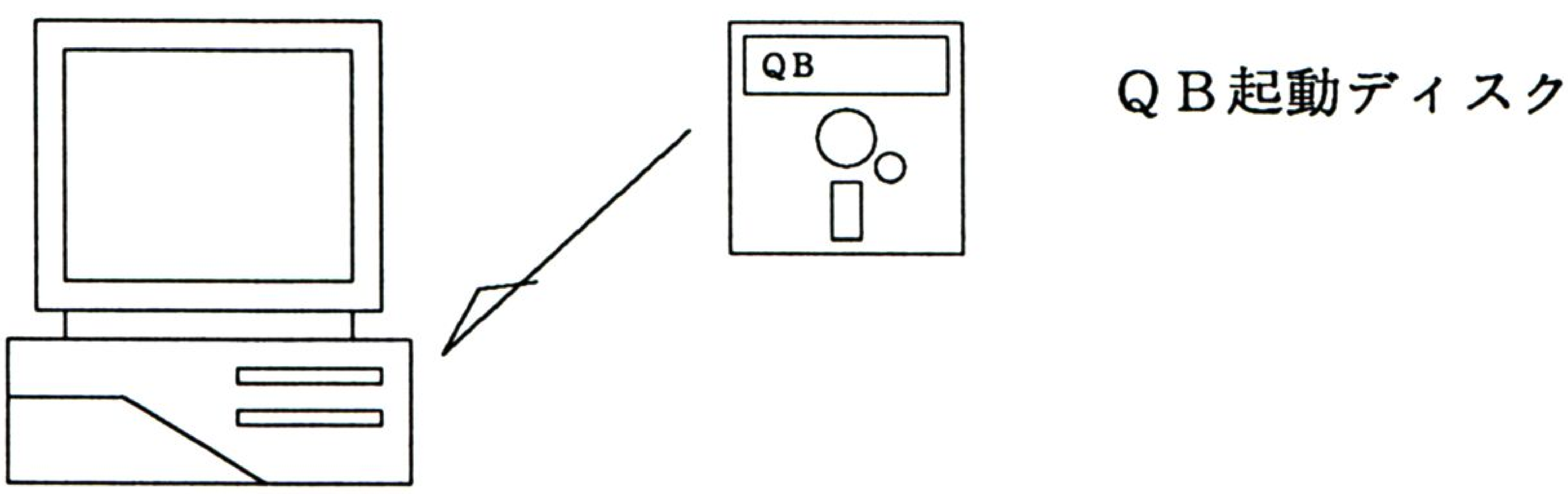



```
A>COPY ATOK*.* B:
ATOK.DIC
ATOK6A.SYS
ATOK6B.SYS
        ３個のファイルをコピーしました.
```

2. ＡドライブへＱＢ起動ディスクをセットし、そのCONFIG.SYSを次のように書き直します。

エディタ（画面上でデータを編集するソフト）を使えば容易にできますが、ない場合は、Ａ＞の画面から次のように入力してください。

改行は ↵ です。また、最後の行の ^Z はファイルの終了の記号で CTRL とＺを同時に押してから、↵ を押します。



```
A>COPY CON CONFIG.SYS
SHELL=A:¥COMMAND.COM A:¥ /P
FILES=20
BUFFERS=10
DEVICE=B:ATOK6A.SYS /T=1 /D=B
DEVICE=B:ATOK6B.SYS
DEVICE=PRINT.SYS
^Z
        1個のファイルをコピーしました.
```

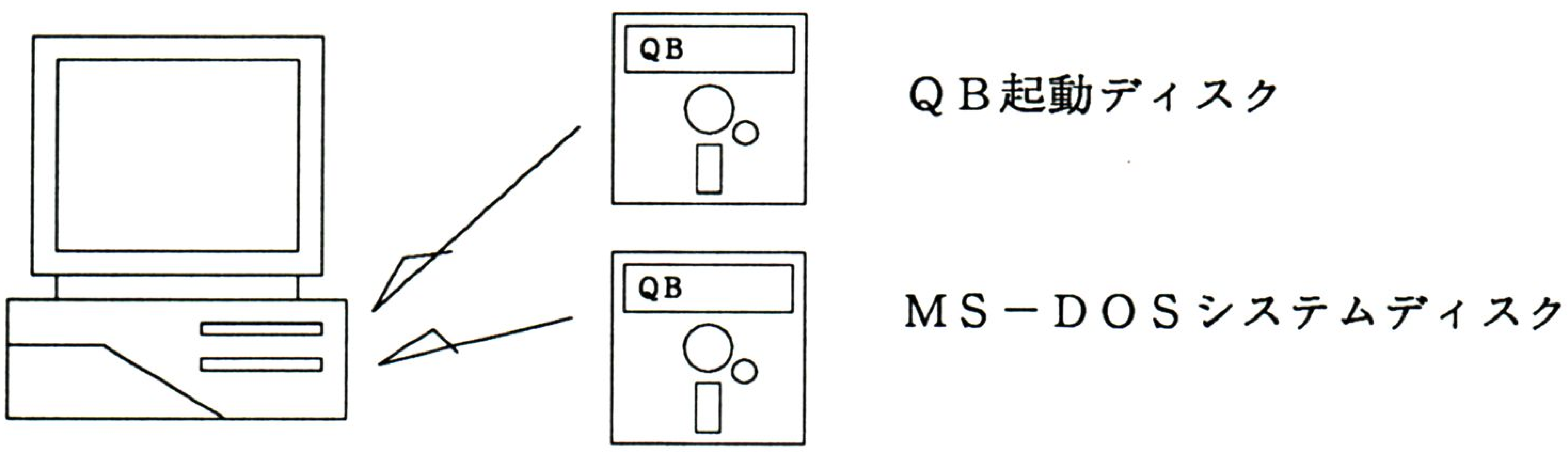
さて、このCONFIG.SYSとは何でしょう。CONFIG はCONFIGURATION（構成）の略で、ユーザーがデバイスドライバなどを追加設定するためのファイルです。

デバイスとは機器という意味で、これを設定することによって各種の機器や漢字変換システムを利用できるようになります。

最後の行にDEVICE=PRINT.SYSという行がありました。これはプリンタドライバの設定で、これによってプリンタを利用できるようになります。

PRINT.SYS はＭＳ－ＤＯＳのシステムディスクにあるので、それを複写します。Ｂドライブへ ＭＳ－ＤＯＳのシステムディスクをセットし、Ａドライブへ ＱＢ起動ディスクをセットして、COPY B:PRINT.SYS A: と入力して ↵ を押します。すると、ＱＢ起動ディスクへPRINT.SYSが複写されます。

これで完成です。




```
A>COPY B:PRINT.SYS A:
        1個のファイルをコピーしました.
```

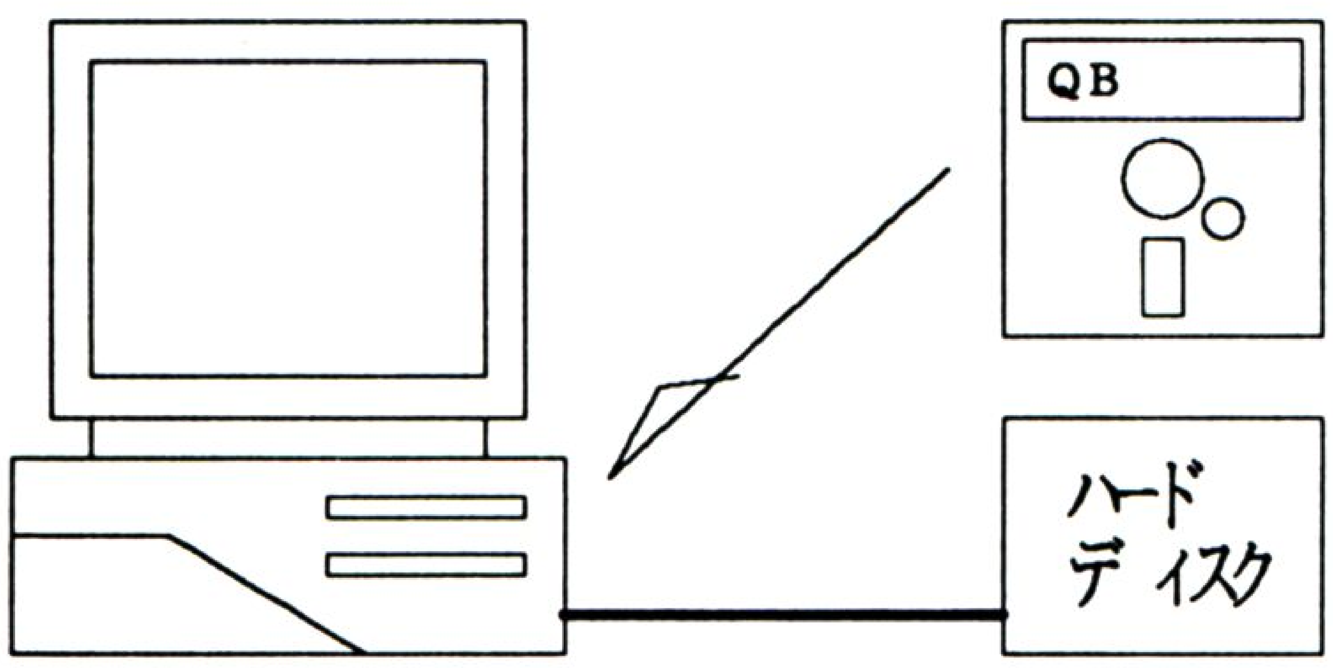
1．4　ハードディスクで使う

Ｑｕｉｃｋ　ＢＡＳＩＣは、ハードディスクに組み込んで使用するのが、最も快適です。

実行用のＱｕｉｃｋ　ＢＡＳＩＣをハードディスクへインストールする手順は次のとおりです。

1. まず、ハードディスクから起動します。
　すると、ハードディスクがＡドライブ、フロッピーディスクがＢドライブとＣドライブになります。

2. ＢドライブにＱｕｉｃｋ　ＢＡＳＩＣの「セットアップディスク」をセットして、B:と入力して ↵ を押し、次にSETUPと入力して ↵ を押します。
　すると、フロッピーディスクへのインストールの場合と同様のメッセージが画面へ表示されます。あとは画面の指示に従って、Ｑｕｉｃｋ　ＢＡＳＩＣの「プログラムディスク」、「ライブラリディスク」、「アドバイザディスク」からハードディスクに必要なファイルを複写してください。普通は、あらかじめ用意されているメニュー選択のまま ↵ を押していけばよいようです。



セットアップディスク

ハードディスクへインストールされる

```
A>B:
B>SETUP
```

第❷章

楽しい入門

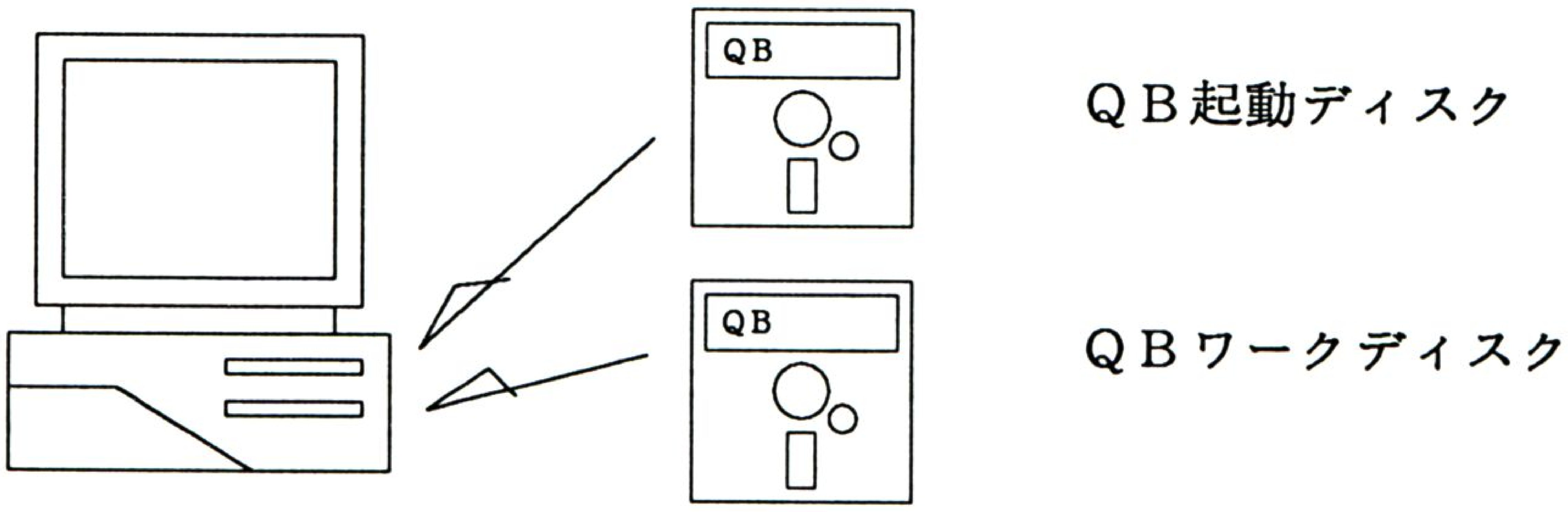
グラフィックスを楽しみながらQBの基本を学びます。

２．１　Ｑｕｉｃｋ　ＢＡＳＩＣの起動

いよいよ、Ｑｕｉｃｋ　ＢＡＳＩＣを始めましょう。「ＱＢ起動ディスク」、「ＱＢワークディスク」及び「ＱＢアドバイザディスク」を用意してください。

1. 電源スイッチを押してＯＮにします。

2. Ａドライブに「ＱＢ起動ディスク」を、Ｂドライブに「ＱＢワークディスク」をセットします。



3. リセットボタンを押します。

4. 画面に表示しながら起動処理が行われ、それが停止したら QB と入力して ↵ を押します。

```
A>PATH=A:¥BIN
A>SET LIB=A:¥LIB
A>SET INCLUDE=B:¥INCLUDE
A>SET TMP=B:¥
A>MOUSE
   --- Installing Microsoft MOUSE Device Driver v6.00 ---
A>QB
```

5. Ｑｕｉｃｋ　ＢＡＳＩＣの統合環境の画面が現れ、ここからプログラムを入力できます。

ＨＥＬＰ画面ならば、ESC を押すと消えます。

Ｑｕｉｃｋ　ＢＡＳＩＣは従前のＢＡＳＩＣと異なって、一つの画面から編集、実行、コンパイル、デバッグなど、ソフト開発のための作業を効率的に行うことができるようになっています。このような開発環境を、統合環境といっています。

２．２　たった１行のプログラム

いよいよ、プログラムの入力です。

LINE(0,0)-(100,100),4,B [↵] と入力してください。ここで最後に [↵] を押すと、適切な位置にスペースが挿入されて、次のように画面に表示されます。

```
LINE (0, 0)-(100, 100), 4, B
```

これでプログラムの入力が終りました。すぐに、実行できます。

プログラムの実行は、[SHIFT] と [f・5] を同時に押すと開始されます。

正方形が描けたでしょう。

今日は、あなたが生まれて初めてＱｕｉｃｋ　ＢＡＳＩＣのプログラムを入力して実行した記念日です。

さて、正方形は描けたのですが、画面上で他の文字と重なっています。そこで、画面上の文字などを消去してから、正方形を描くことにします。

その方法はCLSをLINEの前の行に挿入するだけです。カーソルをLINEのLに合わせて、CLS [↵] を押してください。

```
CLS
LINE (0, 0)-(100, 100), 4, B
```

もう一度実行してください。今度は、文字が消えて赤い正方形が描かれたでしょう。

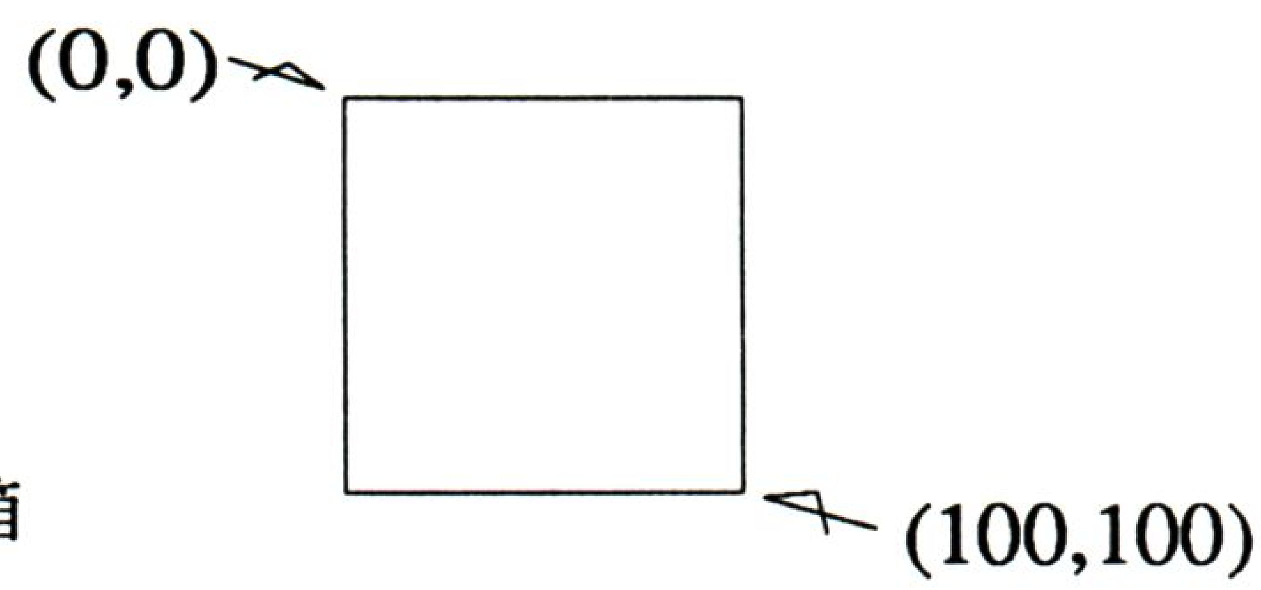
CLSは、CLear Screenの略で、画面を消去するコマンドです。

LINE・・・・は、LINE・・・,Bの組みで、Box（箱）の形にLINEを描きなさいという意味です。

(0, 0)-(100, 100)は、正方形の向い合う頂点の画面上の物理的な位置を座標として表したものです。

4は、色番号で赤のことです。

```
LINE (0, 0)-(100, 100), 4, B
↑                       ↑  ↑
線                      赤  BOX=箱
```



2．3　色はいろいろ

4は色番号で赤を指していますが、ほかにはどのような色を使うことができるのでしょうか。

LINE (0, 0)-(100, 100), 4, Bの中の4を、いろいろな数値に取り替えて実行してください。

4を削除するには、矢印キーまたはマウスでカーソルを4に合わせてから、DEL を押してください。

色の番号は次のとおりです。

0	黒
1	青
2	緑
3	水
4	赤
5	紫
6	黄
7	白

Ｑｕｉｃｋ　ＢＡＳＩＣでは、色番号は0から7までですが、それ以上の数値でもエラーにならないで作動するようになっています。

2．4　小さな箱や大きな箱

(0,0)-(100,100)は、長方形の向い合う頂点の座標を表しています。この数値を変更すると、小さい箱や大きい箱（大小さまざまな正方形や長方形）を描くことができます。

次の下線の部分へ、いろいろな数値を入れて実行してください。

```
LINE(____,____)-(____,____),____, B
```

INS は、「上書きモード」と「挿入モード」を切り替えます。

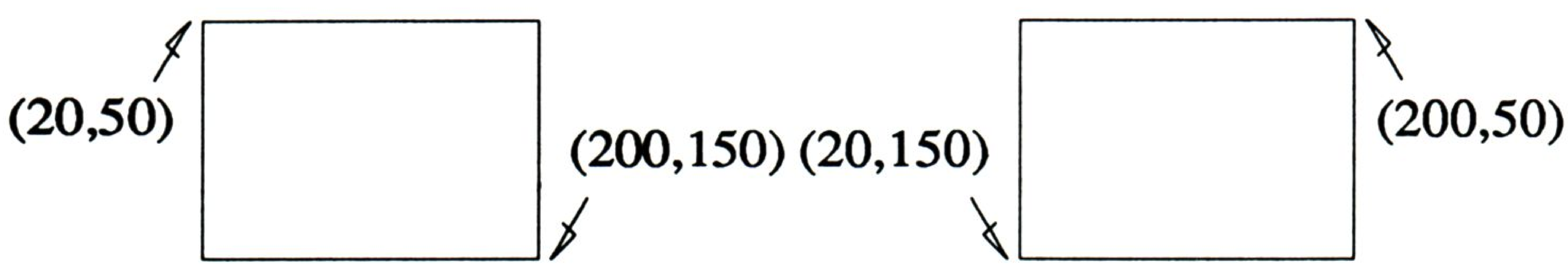
上書きモードでは、修正したい文字の上へカーソルをおいて入力すると、上書きされて、修正前の文字が消されます。

また挿入モードでは、もとの文字は消されないで入力した文字がカーソルの位置へ挿入されます。通常は挿入モードの方が使いよいようです。

数値を置き換えることによって、次のことに気付かれたことでしょう。

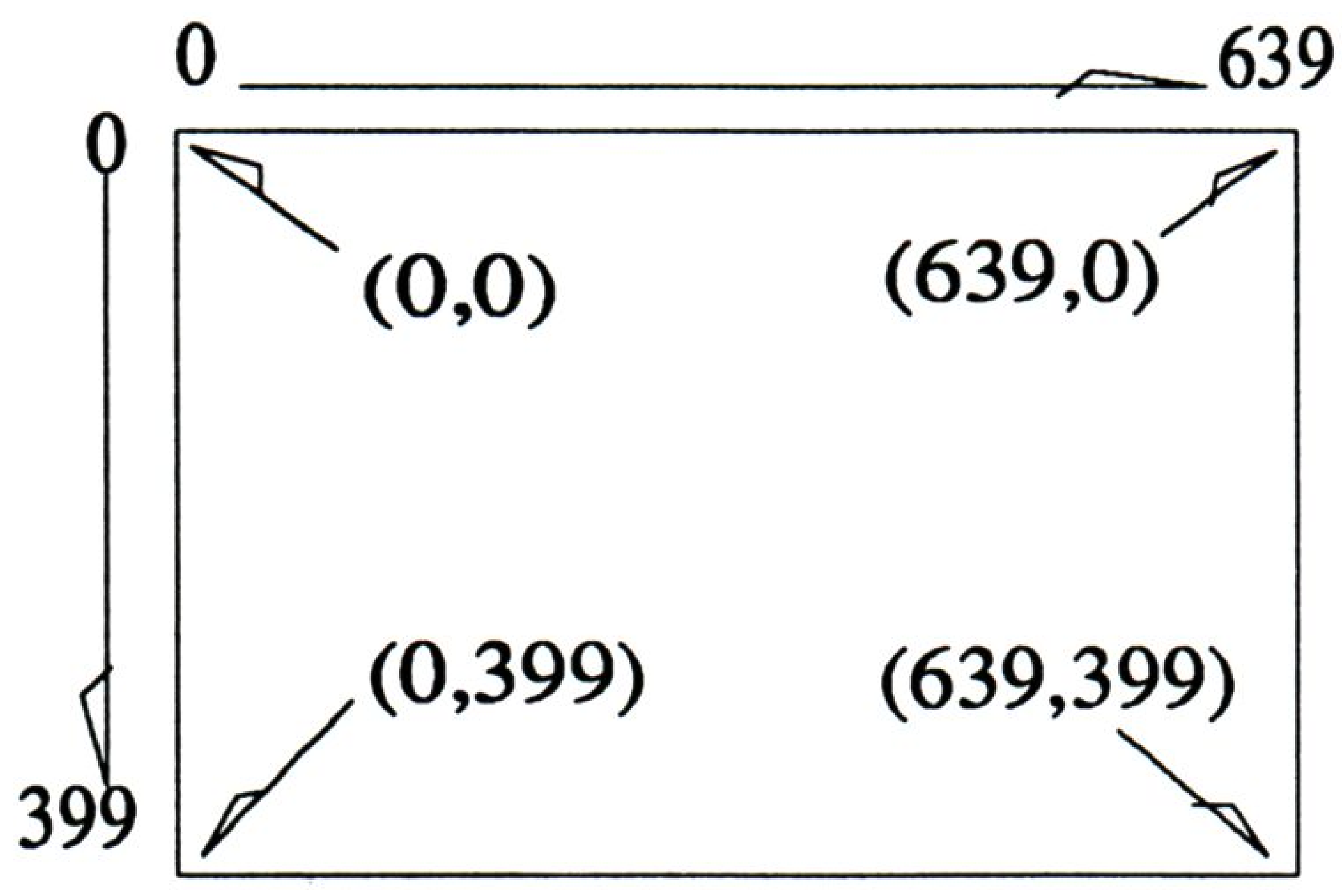
1. 長方形の向い合う頂点の組み合せは、左上と右下の組でも、左下と右上の組でもかまいません。

左上と右下　　　　　　　　　　左下と右上



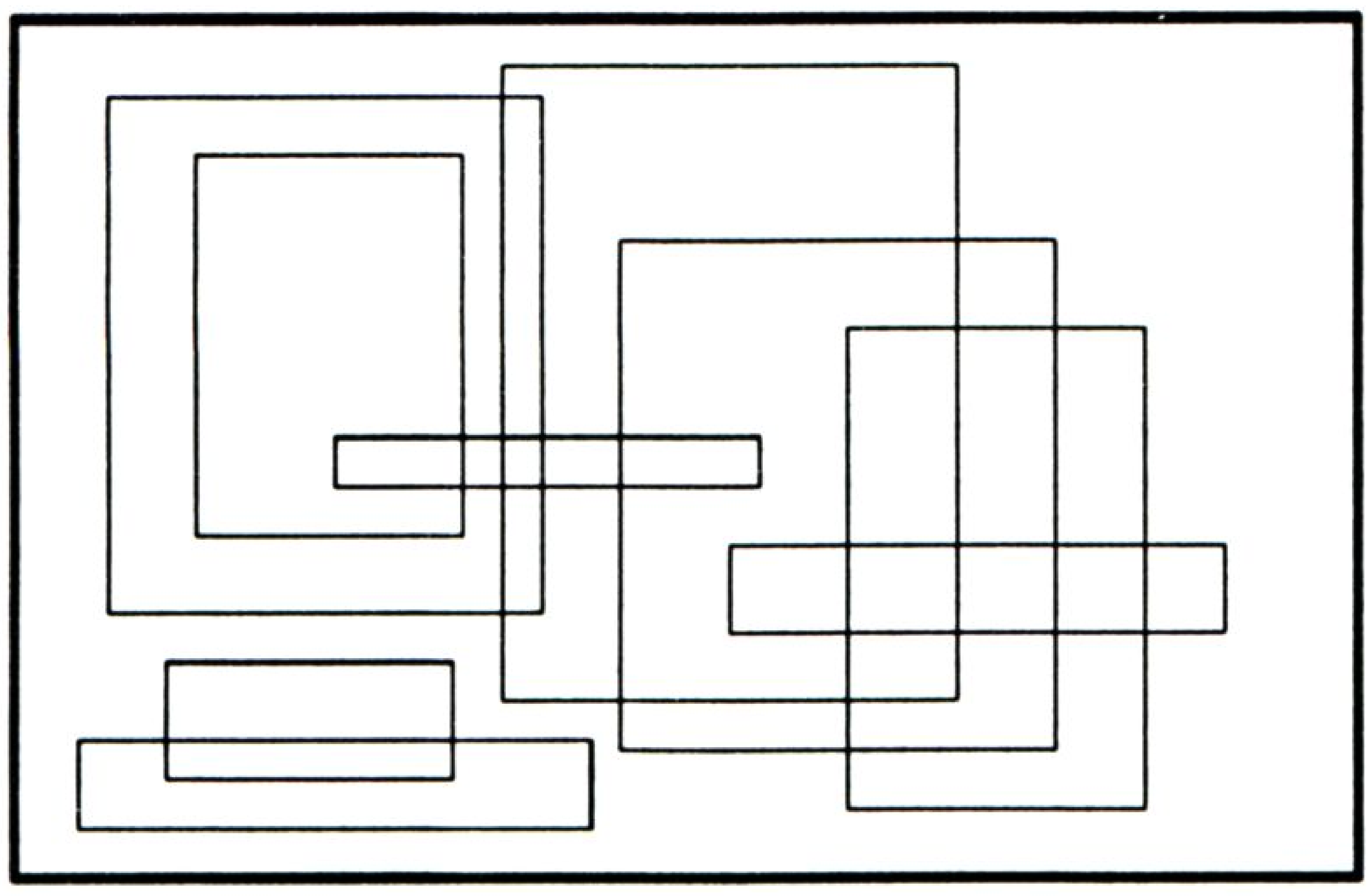
LINE (20, 50)-(200, 150), 4, B　　　LINE (20, 150)-(200, 50), 4, B

2. 座標の数値は、マイナスでも画面より大きくてもエラーになりませんが、画面に描かれるのは、横方向に0から639まで、縦方向に0から399までです。



2．5　複写で箱の芸術

LINEステートメントでいろいろな箱を描いて、「箱を主題とするコンピュータグラフィックス」を創作します。



このようなプログラムをつくるには、LINEステートメントを何回も繰り返して入力する必要があります。そこで、LINEを複写して数値のみを書き換えることによって比較的楽に入力することができます。

複写する方法は次のとおりです。

1. 複写する範囲を指定します。

複写を開始する行にマウスカーソルを合わせて、マウスの左ボタンを押したまま終了行まで移動すると、複写元の範囲の行が反転して表示されます。

マウスを使わない場合は、複写開始行にカーソルを合わせて、複写終了行まで SHIFT を押しながら矢印キーを押していくと、複写元の範囲の行が反転して表示されます。

2. バッファーに複写元の内容を読み込んで記憶します。

CTRL を押しながら INS を押します。

3. バッファーに記憶された複写内容を複写先の位置へ貼り付けます。

複写先の開始位置へカーソルを合わせて、SHIFT を押しながら INS を押します。貼り付けが終ってもバッファーに記憶した内容はそのままですから、何度も繰り返して貼り付けることができます。

```
CLS
LINE(0, 0)-(100, 100), 4, B
LINE(20, 520)-(1200, 10), 5, B
LINE(180, 250)-(152, 510), 6, B
LINE(-10, -500)-(400, 300), 3, B
LINE(300, 30)-(380, 370), 2, B
```

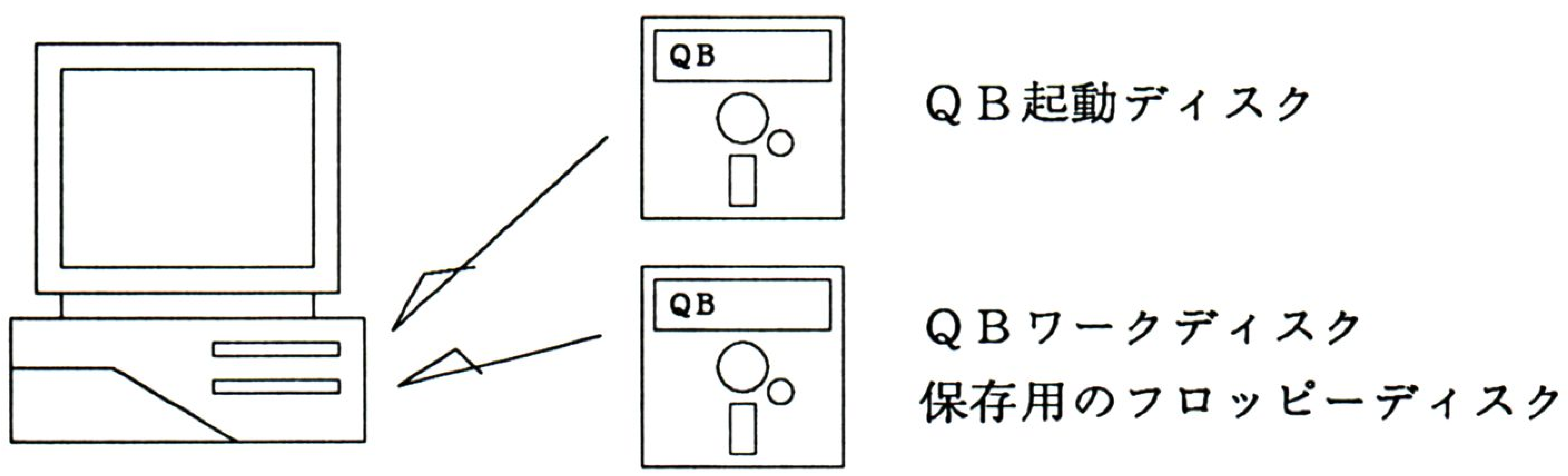
2．6　プログラムの保存

苦心して創作した「箱の芸術」のプログラムも、電源を切ると即座に消えてしまいます。

そこで、プログラムをフロッピーディスクへ保存します。

1． 保存するフロッピーディスクをＢドライブへセットします。

　このフロッピーディスクはフォーマット済みのものです。「ＱＢワークディスク」に保存しても差し支えありませんが、残りの容量があまり多くはないのですぐにディスクがいっぱいになってしまいます。



2． 画面左上の［F/ファイル］にマウスカーソルを合わせて、左クリックします。

　左クリックとは、マウスの左ボタンを押すことです。

　マウスを使用しないときは、GRPH を押してからＦを押して［F/ファイル］を選択ます。

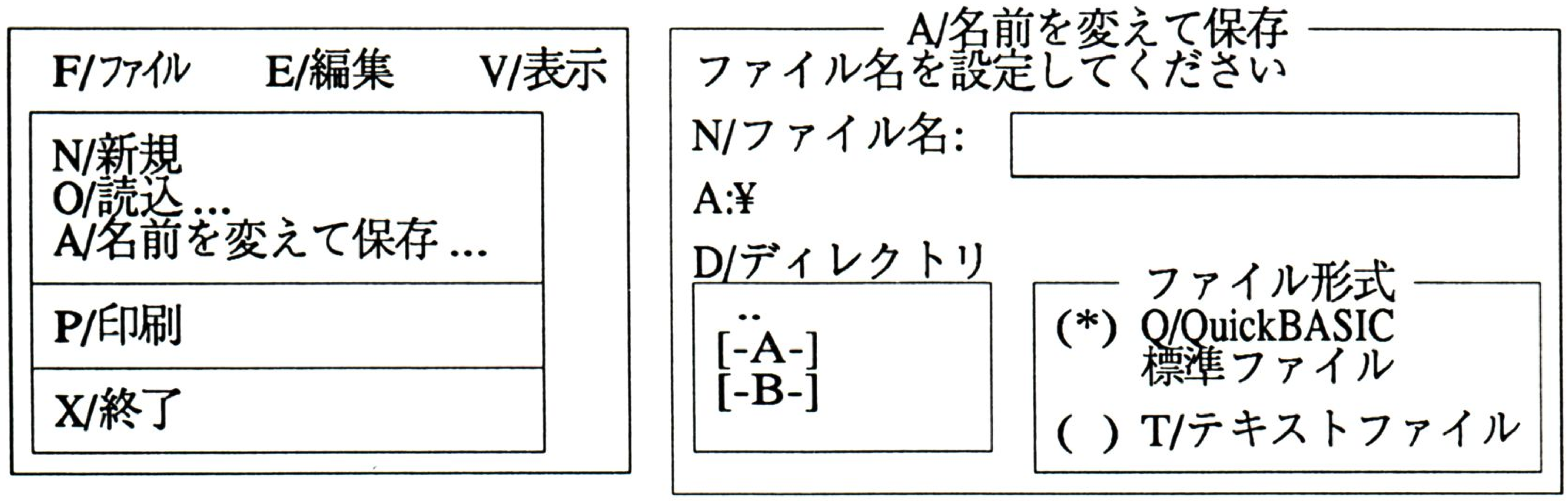
　すると、［F/ファイル］のメニューが表示されます。

3． ［F/ファイル］のメニューの［A/名前を変えて保存］を左クリックします。

　マウスを使用しない場合は、Ａを押します。

4． ［A/名前を変えて保存］のメニューが表示され、「ファイル名を設定してください」と指示されるので、例えばB:QB2_5　と入力します。

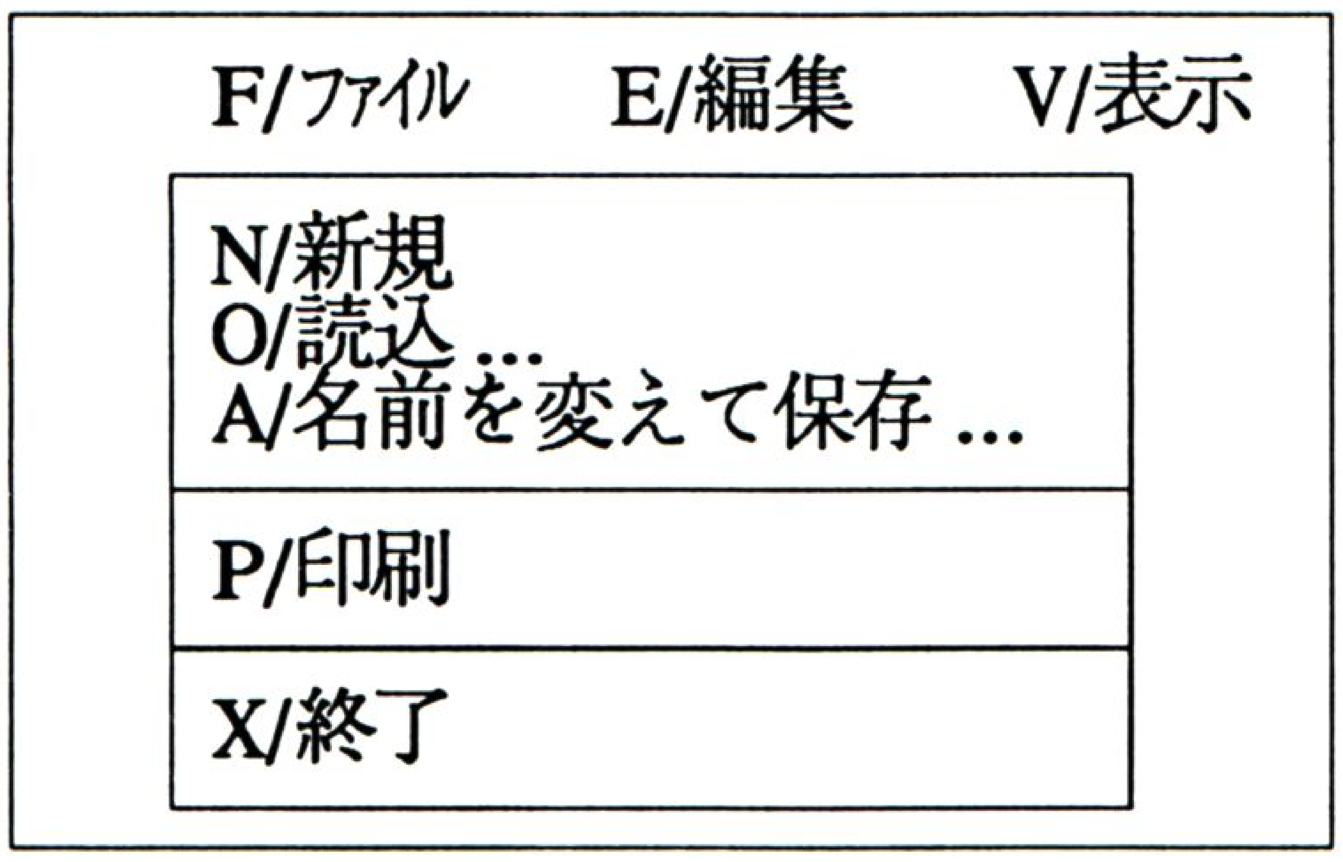
　B:はＢドライブということです。QB2_5はプログラム名で第２章の５節の例題ということがわかるような名前にしました。プログラム名は、英数字８文字までの範囲内で作者が自由に付けることができます。



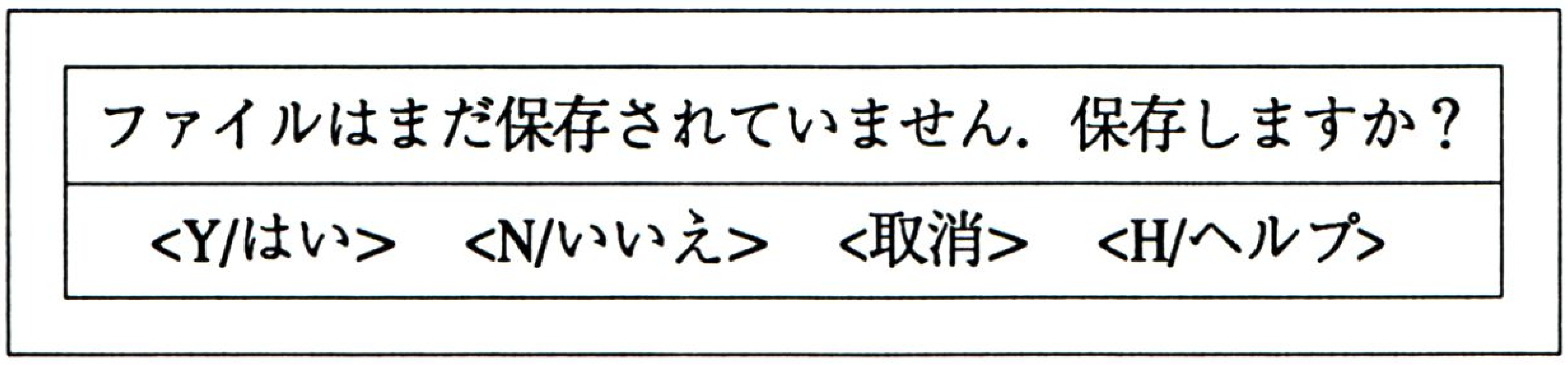
２．７　Ｑｕｉｃｋ　ＢＡＳＩＣの終了

Ｑｕｉｃｋ　ＢＡＳＩＣの終了は次の手順で行います。

1. ［F/ﾌｧｲﾙ］を左クリックします。あるいは GRPH を押してＦを押します。
すると、［F/ﾌｧｲﾙ］のメニューが表示されます。



2. ［X/終了］を左クリックします。あるいはＸを押します。
ファイルが保存されていない場合は、確認のメニューが表示されます。



3. 現在編集中のプログラムを保存する場合は［Y/はい］を、保存しない場合は［N/いいえ］を左クリックします。あるいはＹまたはＮを押します。
保存する場合は、保存のメニュー画面が表示されるので、保存します。
保存しない場合は、直ちに終了し、取消の場合は、編集画面に戻ります。

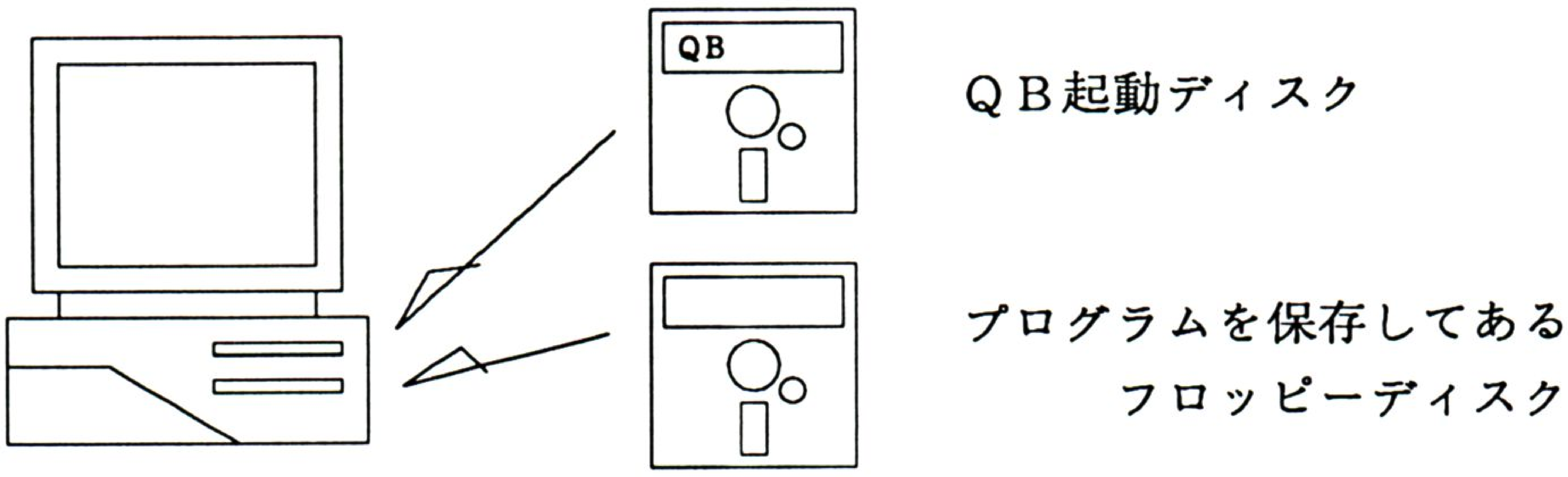
4. Ｑｕｉｃｋ　ＢＡＳＩＣの編集画面が消えてＭＳ－ＤＯＳレベルの画面が表示されたら、フロッピーディスクを取り出します。

5. 電源を切ります。

2．8　プログラムの読み込み

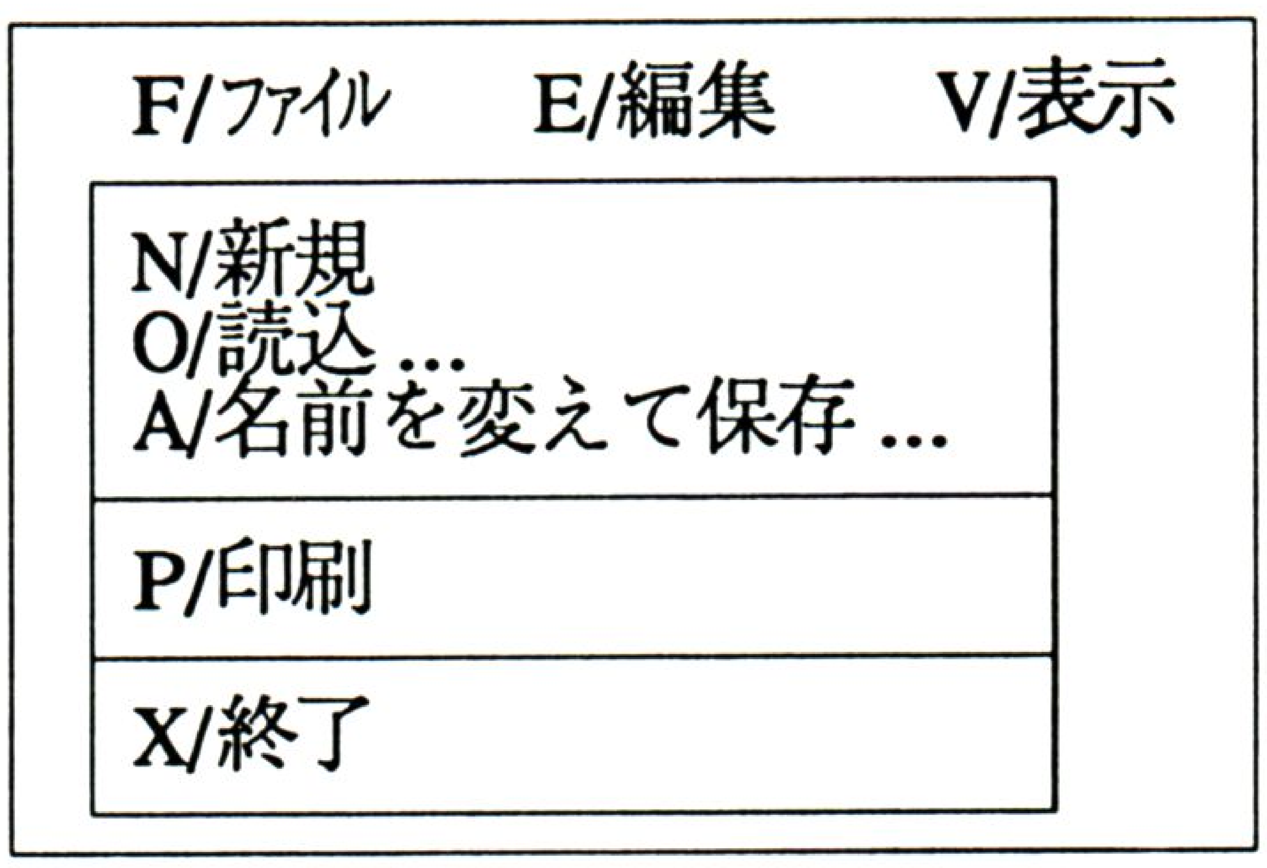
フロッピーディスクに保存してあるプログラムを読み込む手順は次のとおりです。

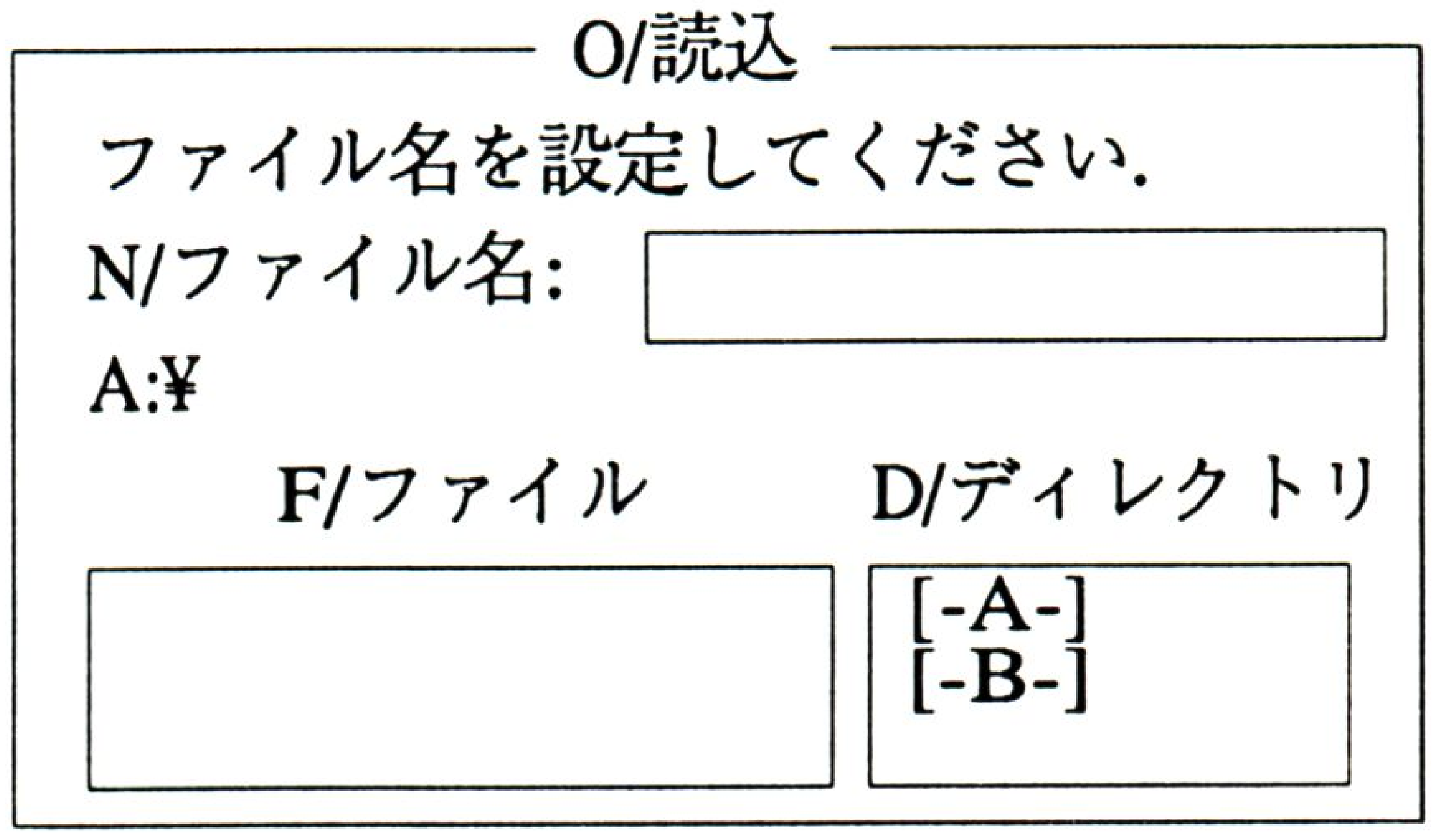
1. プログラムを保存してあるフロッピーディスクをＢドライブにセットします。



2. ［F/ファイル］を左クリックします。あるいは GRPH を押してＦを押します。
すると、［F/ファイル］のメニューが表示されます。

3. ［O/読込］を左クリックします。あるいはＯを押します。
すると、［O/読込］のメニューが表示されます。





4. ［D/ディレクトリ］の欄の[-B-]を左ダブルクリックします。
ダブルクリックは、マウスのボタンを２度続けて押すことです。カチカチと素早く押してください。
マウスを使用しない場合は、 TAB を押してカーソルを［D/ディレクトリ］の欄へ移動し、矢印キーを押して[-B-]を反転させて ↵ を押します。
すると、［F/ファイル］の欄にＢドライブに保存されているプログラムの名前の一覧が表示されます。

5. ［F/ファイル］の欄に表示されているプログラム名一覧の中から、読み込むプログラム名を左ダブルクリックします。マウスを使用しない場合は、 TAB と矢印キーでプログラムを選択して ↵ を押します。
すると、指定したプログラムが読み込まれて、編集画面に戻ります。

2．9　箱に色を塗りましょう

正方形や長方形などいろいろな箱を描いて、箱の芸術ができましたか？
これまでの箱では、箱の中が黒色でした。そこで、中に色を塗ることにします。

箱に色を塗るには、LINEステートメントのBにFを加えて、BFとすればよいのです。

```
LINE (10, 20)-(150, 350), 4, BF
```

Fは、満たすという意味のFILLからきています。BOXのBと合わせて、BFは箱の中を満たすということです。

どうぞ、LINEステートメントのB及びBFを使用して、「箱の芸術作品」を作ってください。

作例

```
CLS
LINE (20, 10)-(200, 100), 4, B
LINE (-200, 110)-(250, -100), 5, BF
LINE (120, 156)-(600, 390), 6, BF
LINE (220, 10)-(100, 370), 7, BF
LINE (280, 210)-(630, 375), 1, BF
LINE (150, 30)-(225, 340), 2, BF
LINE (420, 10)-(540, 291), 3, BF
LINE (482, 50)-(620, 350), 4, B
LINE (330, 130)-(230, 400), 5, B
LINE (20, 10)-(200, 100), 6, B
LINE (220, 150)-(320, 460), 5, B
```

２．１０　４行で１００個の箱を描く

次のプログラムは、どのような図を描くと思いますか。

```
CLS
LINE (0, 0)-(1, 1), 1, B
LINE (0, 0)-(2, 2), 2, B
LINE (0, 0)-(3, 3), 3, B
LINE (0, 0)-(4, 4), 4, B
    ..................
    .....    省略    .....
    ..................
LINE (0, 0)-(97, 97), 97, B
LINE (0, 0)-(98, 98), 98, B
LINE (0, 0)-(99, 99), 99, B
LINE (0, 0)-(100, 100), 100, B
```

このプログラムは、箱の左上の頂点は(0,0)で右下の頂点が(1,1)から(100,100)まで変化して、１００個の箱を描きます。

ところが、ＱｕｉｃｋＢＡＳＩＣには便利な道具があって、上の１０１行ものプログラムをたった４行で書くことができます。

```
CLS
FOR I = 1 TO 100
    LINE (0, 0)-(I, I), I, B
NEXT I
```

この道具は、FORとNEXTと間のプログラムを、Iを変化させながら繰り返し実行する仕掛けです。

```
FOR I = 1 TO 100
  （Iを1から100まで変化させながら
   １００回繰り返す）
NEXT I
```

実行の経過を詳しく示すと、次のようになります。

１回目	I=1で実行	LINE (0, 0)-(1, 1), 1, B
２回目	I=2で実行	LINE (0, 0)-(2, 2), 2, B
３回目	I=3で実行	LINE (0, 0)-(3, 3), 3, B
４回目	I=4で実行	LINE (0, 0)-(4, 4), 4, B
	（中間を省略）	
９７回目	I=97で実行	LINE (0, 0)-(97, 97), 97, B
９８回目	I=98で実行	LINE (0, 0)-(98, 98), 98, B
９９回目	I=99で実行	LINE (0, 0)-(99, 99), 99, B
１００回目	I=100で実行	LINE (0, 0)-(100, 100), 100, B

ところで、このプログラムの中で使用しているI（アイ）とは何でしょう。

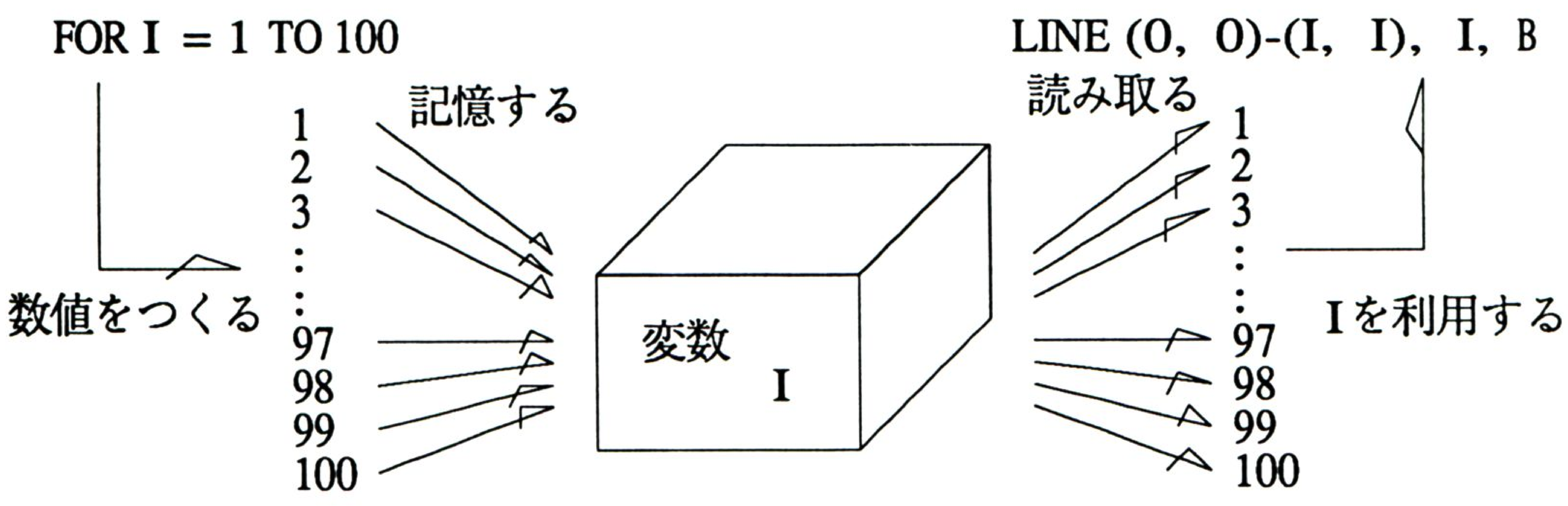
Iは、数値を記憶する容器のようなものです。

FOR....NEXTは、Iという名前の容器に1から100までの数値を順に入れて、その I の内容を使ってLINE...を繰り返し実行させているのです。

Iのように数値を入れて使用する容器のようなものを、「変数」といいます。しかし、「変数」といっても数学でいう「変数」とは異ります。また、変数にもいろいろな種類があるのですが、このことはあとで説明します。

さて、変数の名前は I（アイ）だけなのでしょうか。

変数の名前は、プログラムを作るときによく考えて、わかりやすいように付けます。変数名を上手に付けないと、プログラムを修正したりエラーを発見する場合に、たいへんな難儀をすることもあります。特に複数の担当者が共同開発する場合は、変数名の付け方について、共通理解が必要になります。



2．11　等間隔の数値をつくる

次のプログラムを読んでください。

```
CLS
LINE (0, 0)-(10, 10), 10, B
LINE (0, 0)-(20, 20), 20, B
LINE (0, 0)-(30, 30), 30, B
LINE (0, 0)-(40, 40), 40, B
LINE (0, 0)-(50, 50), 50, B
LINE (0, 0)-(60, 60), 60, B
LINE (0, 0)-(70, 70), 70, B
LINE (0, 0)-(80, 80), 80, B
LINE (0, 0)-(90, 90), 90, B
LINE (0, 0)-(100, 100), 100, B
```

箱の頂点の位置が10,20,30,40,50,60,70,80,90,100というように、10ずつ増加していることに気付くでしょう。

このプログラムを、FOR.....NEXTを利用して書くと次のようになります。

```
CLS
FOR I = 1 TO 100 STEP 10
    LINE (0, 0)-(I, I), I, B
NEXT I
```

FOR I = 1 TO 100 STEP 10 は、Iは10から100まで10の間隔で増加しながら繰り返すということです。

STEPは、増加、減少の間隔を表しており、プラスの数値ならば増加、マイナスの数値ならば減少します。

例　　FOR A = 5 TO 200 STEP 4

　　　　Aは、5から100まで4の間隔で増加して繰り返します。

　　FOR B = 80 TO -150 STEP -5

　　　　Bは、80から-150まで5の間隔で減少して繰り返します。

上のプログラムのうち、FOR I = 1 TO 100 STEP 10の下線の部分を、いろいろな数値に取り替えて実行してください。

STEPの数値が大きい場合と小さい場合とでは、どのように画面の箱の様子が変わるでしょうか。

2．１２　もっと縞模様を

箱を重ねて描くことによって、縞模様ができました。そこで今度は、縞模様を画面の他の位置にも多数描くようにします。そのためには、箱の左上の頂点の位置を変化させればよいのです。

```
'QB2_12_1
CLS
FOR I = 0 TO 100
        LINE (0, 0)-(I, I), I, B
NEXT I
FOR I = 50 TO 150
        LINE (50, 50)-(I, I), I, B
NEXT I
FOR I = 100 TO 200
        LINE (100, 100)-(I, I), I, B
NEXT I
FOR I = 150 TO 250
        LINE (150, 150)-(I, I), I, B
NEXT I
```

'QB2_12_1 は、このプログラムの名前です。わかりやすいように章と節の番号を使いました。先頭に　'がありますが、これはこの行が注釈行（説明のための部分）であることを示しています。_ は、アンダースコア（下線）といいます。コボルでは -（マイナス、ハイフン）を使うことが多いようですが、ＢＡＳＩＣやＣ言語ではアンダースコアの方が使いやすいようです。

1． 箱の左上の頂点の位置の変化に注目してください。

(0,0)、(50,50)、(100,100)、(150,150)と、50間隔で変化しています。そこで、この変化する数値を Jという名前の変数に入れて使うことにします。

```
FOR J = 0 TO 150 STEP 50

NEXT J
```

2. FOR I = ...NEXT Iが４組あります。

開始の値が、0,50,100,150と 0から 150まで50ずつ増加しています。これはJ の変化と同じです。終りの値は開始の値に 100を加えた数になっています。すなわち、J+100ということです。そこで、次のようにかくことができます。

```
FOR I = J TO J + 100


NEXT I
```

3. 最後に、１．の中に２．を組み込むと次のようになります。

```
'QB2_12_2
FOR J = 0 TO 150 STEP 50
    FOR I = J TO J + 100
        LINE (J, J)-(I, I), I, B
    NEXT I
NEXT J
```

このように、FOR....NEXTの中にFOR....NEXTが入る形を「入れ子」といいます。

入れ子のプログラムをつくることによって、LINEの中の数値をいろいろに変化させることができました。

次のプログラムの下線の部分の数値を書き換えて、縞模様のプログラムをつくってください。

```
'QB2_12_3
FOR J = 0 TO 150 STEP 50
    FOR I = J TO J + 100
        LINE (J, J)-(I, I), I, B
    NEXT I
NEXT J
```

2．13　大から小への変化

これまでは、最初は小さい箱を描き、次第に大きい箱を繰り返し描いて縞模様をつくりました。今度は、最初に大きい箱を描き、次第に小さい箱を繰り返し描いて縞模様をつくることにします。

次第に大きくする場合の例

FOR I ＝ 0 TO 100 STEP 5　　　　Iは0から100まで5ずつ増加します。

次第に小さくする場合の例

FOR I ＝ 100 TO 0 STEP -5　　　　Iは100から0まで5ずつ減少します。

増加のプログラムの例

```
CLS
LINE (0, 0)-(0, 0), 0, B
LINE (0, 0)-(5, 5), 5, B
LINE (0, 0)-(10, 10), 10, B
LINE (0, 0)-(15, 15), 15, B
           :
           :
LINE (0, 0)-(95, 95), 95, B
LINE (0, 0)-(100, 100), 100, B
```

減少のプログラムの例

```
CLS
LINE (0, 0)-(100, 100), 100, B
LINE (0, 0)-(95, 95), 95, B
LINE (0, 0)-(80, 80), 80, B
LINE (0, 0)-(85, 85), 85, B
           :
           :
LINE (0, 0)-(5, 5), 5, B
LINE (0, 0)-(0, 0), 0, B
```

［FOR....NEXTを使う増加］

```
CLS
FOR I = 0 TO 100 STEP 5
    LINE (0, 0)-(I, I), I, B
NEXT I
```

［FOR....NEXTを使う減少］

```
CLS
FOR I = 100 TO 0 STEP -5
    LINE (0, 0)-(I, I), I, B
NEXT I
```

上のプログラムを実行してください。よく見ると違う部分がありますが、ほとんど同じ結果になっています。ところが、LINE....の最後の BをBFに替えて実行すると、まったく違う結果になります。

そこで、QB2_12_2のプログラムを次のように書き直して、少し幅の広い縞模様にします。

1. 大の箱を先に小の箱を後に描くようにします。

J TO J + 100 を　J + 100 TO J　とします。

2. 5ずつ小さい箱を描くようにします。

STEP -5 を加えます。

3. BをBFへ描きなおします。

BへFを加えて、BFとします。

改造前のプログラム

```
'QB2_12_2
CLS
FOR J = 0 TO 150 STEP 50
    FOR I = J TO J + 100
        LINE (J,  J)-(I,  I),  I,  B
    NEXT I
NEXT J
```

改造後のプログラム

```
'QB2_13_1
CLS
FOR J = 0 TO 150 STEP 50
    FOR I = J + 100 TO J STEP -5
        LINE (J,  J)-(I,  I),  I, BF
    NEXT I
NEXT J
```

2．14　キーボードからの入力

これまで描いた模様は、画面の左上すみから斜めに表示されています。

そこで、模様を描く位置を全体に 200だけ右に移動したいときは、横座標の数値だけを 200ずつ増やして、(J+200,J)と(I+200,I)とすればよいのです。

```
'QB2_14_1
CLS
FOR J = 0 TO 150 STEP 50
    FOR I = J + 100 TO J STEP -5
      LINE (J + 200, J)-(I + 200, I), I ,BF
    NEXT I
NEXT J
```

次に、縞模様を描く位置を、キーボードから入力して表示するように改造したいと思います。キーボードから入力すれば、プログラムを書きなおさないで表示する位置を変えることができます。

```
'QB2_14_2
CLS
INPUT "ITI=";ITI
FOR J = 0 TO 150 STEP 50
    FOR I = J + 100 TO J STEP -5
        LINE (J + ITI, J)-(I + ITI, I), I ,BF
    NEXT I
NEXT J
```

INPUTは、キーボードからデータを入力するステートメントです。

INPUTの後ろの"ITI="は、画面上にこれから何を入力するかを案内する言葉です。これがないとパソコンを使う人が、次に何をしたらよいか分らなくなってしまうのです。ですから、"ｲﾁｦ ｵﾈｶﾞ ｲｼﾏｽ･･･"でも、"位置を入力してください-->"でもよいのです。

位置のデータを入れる変数の名をITIとしました。INPUTステートメントの最後のITIは、キーボードで入力した数をITIという変数に入れることです。

これを実行すると、画面に　ITI=　と表示されます。ここでキーボードから 200と入力します。すると、ITIという変数に200が入ります。

以後は、容器のITIをプログラムの中で使えるのです。ITIへ入力する数は、300でも400でもまた-100でもよいのです。いろいろ繰り返して実行してください。

では次に、どのような幅の縞模様がよいかを試行するために、縞の幅をキーボードから入力するプログラムを作りましょう。

```
'QB2_14_3
CLS
INPUT "HABA=";HABA
FOR J = 0 TO 150 STEP 50
    FOR I = J + 100 TO J STEP HABA
        LINE (J, J)-(I, I), I ,BF
    NEXT I
NEXT J
```

変数HABAの数値がマイナスになっていなければ画面には何も表示されません。つねにマイナスの入力を求めるのでは不便なので、

FOR I = J + 100 TO J STEP HABA * -1

と改造します。つねに-1を掛けるので、プラスの値を入力すればSTEPの値はマイナスになります。

さらに、プラスでもマイナスでも必ずマイナスの数にするために、次のように改造します。

```
'QB2_14_4
CLS
INPUT "HABA=";HABA
IF HABA > 0 THEN HABA = HABA * -1
FOR J = 0 TO 150 STEP 50
    FOR I = J + 100 TO J STEP HABA
        LINE (J, J)-(I, I), I ,BF
    NEXT I
NEXT J
```

IF HABA > 0 THEN HABA = HABA * -1　　は、もしも、HABAがプラス（0より大）ならば、HABAへ-1を掛けた結果をHABAへ入れます。プラスの時だけ-1を掛けるので、HABAは、いつもマイナスになります。

2．15　画面いっぱいに描こう

これまで箱でつくった縞模様のテストを繰り返してきましたが、今度は画面いっぱいに描くプログラムをつくりましょう。

ここには、ひとつのプログラムの例を示します。

```
'QB2_15_1
CLS
FOR ITI = -400 TO 600 STEP 100
    FOR J = 0 TO 350 STEP 50
        FOR I = J + 100 TO J STEP -5
            LINE (J + ITI, J)-(I + ITI, I), I ,BF
        NEXT I
    NEXT J
NEXT ITI
```

画面の外にまで描いている無駄なところの多いプログラムですが、単純にするためにやむを得ずこのようにしました。
-400 TO 600 STEP 100 としたのは、斜めに箱を描くようになっていますので、画面の左下に描くには、画面の左の外側か描き始める必要があったからです。

次のプログラムの下線の部分にいろいろな数値を入れて、箱の大きさや縞の幅を変えて実行してください。

```
'QB2_15_2
CLS
FOR ITI =～～～TO～～～STEP～～～
    FOR J =～～～TO～～～STEP～～～
        FOR I = J +～～～TO J STEP～～～
            LINE (J + ITI, J)-(I + ITI, I), I ,BF
        NEXT I
    NEXT J
NEXT ITI
```

２．１６　文字の画面表示と印刷

これまで、数値を指定した間隔で自動的に作りながら繰り返す機能をもっている FOR....NEXTによって縞模様をつくってきました。

では、この作られた数値を画面上に表示するプログラムを実行して、その動きを確認しましょう。

例１　100から500まで50間隔で増加する数をつくります。

```
'QB2_16_1
CLS
FOR I = 100 TO 500 STEP 50
    PRINT I
    LPRINT I
NEXT I
```

PRINTは、画面に表示するステートメントで、PRINT I とすると変数Iの内容の数値を表示します。

LPRINTは、プリンタに印刷するステートメントで、LPRINT I とすると変数Iの内容の数値を印刷します。

このプログラムは、画面およびプリンタに次のように表示、印刷します。

```
100
150
200
250
300
350
400
450
500
```

例２　Jは2000から4000まで1000間隔で増加し、同時にIは100から500まで200間隔で増加します。

```
'QB2_16_2
FOR J  = 2000 TO 4000 STEP 1000
        FOR I  = 100 TO 500 STEP 200
              PRINT  "J=";J,"I=";I
              LPRINT  "J=";J,"I=";I
        NEXT I
NEXT J
```

IとJの数を、PRINT I,J　として表示できますが、IとJが区別できるように"J="と"I="を表示するようにしました。

"J="とJの間には、セミコロン［;］が使われ、Jと"I="との間にはカンマ［,］が使われているのは、間隔の空け方に違いがあるからです。

とりあえず、セミコロンを使えば間隔をとらないし、カンマを使えば十分に間隔をとると覚えておいてください。

このプログラムは、次のように表示、印刷します。

```
J=2000        I=100
J=2000        I=300
J=2000        I=500
J=3000        I=100
J=3000        I=300
J=3000        I=500
J=4000        I=100
J=4000        I=300
J=4000        I=500
```

ループの外側にあるJがゆっくりと変化し、内側のIが忙しく変化することが分ります。

2．17　プログラムの印刷

プログラムを印刷する方法を説明します。

1. ［F/ﾌｧｲﾙ］をマウスで左クリックします。あるいは、GRPH を押し、続いてＦを押します。
　すると、［F/ﾌｧｲﾙ］のメニュー画面が表示されます。

F/ﾌｧｲﾙ　　E/編集　　V/表示

N/新規
O/読込 ...
A/名前を変えて保存 ...

P/印刷

X/終了

2. ［P/印刷］をマウスで左クリックします。あるいはＰを押します。
　すると、次のような［P/印刷］のメニュー画面が表示されます。

P/印刷

印刷する範囲を指定してください
( ) S/ 指定範囲
( ) W/ 現在のウインドウ
(*) M/ 現在のモジュール

<確認>　<取消>　<H/ヘルプ>

3. ＜確認＞をマウスで左クリックします。あるいは、↵ を押します。
　すると、プログラムの印刷が開始します。

印刷されない場合は、プリンタの電源、用紙のセット、プリンタケーブルの接続などを確認してください。

2．18　いろいろな箱の模様

これまではFOR....NEXTで箱の位置や大きさなどを指定して描きましたが、今度は、もっと奔放にいろいろな大きさの箱を描くことにしましょう。

そのために、便利な道具があります。それは RNDという魔法の杖です。この杖を使うと、0（ゼロ）から1（イチ）までの間の数をバラバラに出してくれます。

魔法の杖がどんな数を出すか、次のプログラムで調べましょう。

```
'QB2_18_1
CLS
FOR I = 1 TO 100
     PRINT RND;
NEXT I
```

これを実行すると、次のような数値が画面へ表示されます。

```
.7055475 .533424 .5795186 .2895625 .301948 .7747401
1.401764E-02 .7607236 .81449 .7090379 4.535276E-02
.4140327 .8626193 .79048 .3735362 .9619532 .8714458
5.623686E-02 .9495566 .3640187 .5248684 .7671117
5.350453E-02 .5924582 .4687001 .2981654 .6226967
          （以下　省略）
```

画面に表示された数を観察してください。 RNDは 0より大きく 1より小さい数値を 1と 0の間で均等に出すようにつくられているのです。これを一様乱数といいます。

PRINT RND;の最後に付けた ；（セミコロン）には改行しないという機能があるので、1行にたくさんの数が表示されました。もしも ;を付けず PRINT RND だけならば、数を表示するたびに改行するので1行に1つの数を表示することになります。セミコロンを削除して実行してみてください。

さて、何度も実行すると、そのたびに同じ数の並びが表示されることに気付かれたことでしょう。

実行するたびに異なった乱数が表示されるように、次の仕掛けを加えます。

```
'QB2_18_2
RANDOMIZE TIMER
CLS
FOR I = 1 TO 100
    PRINT RND
NEXT I
```

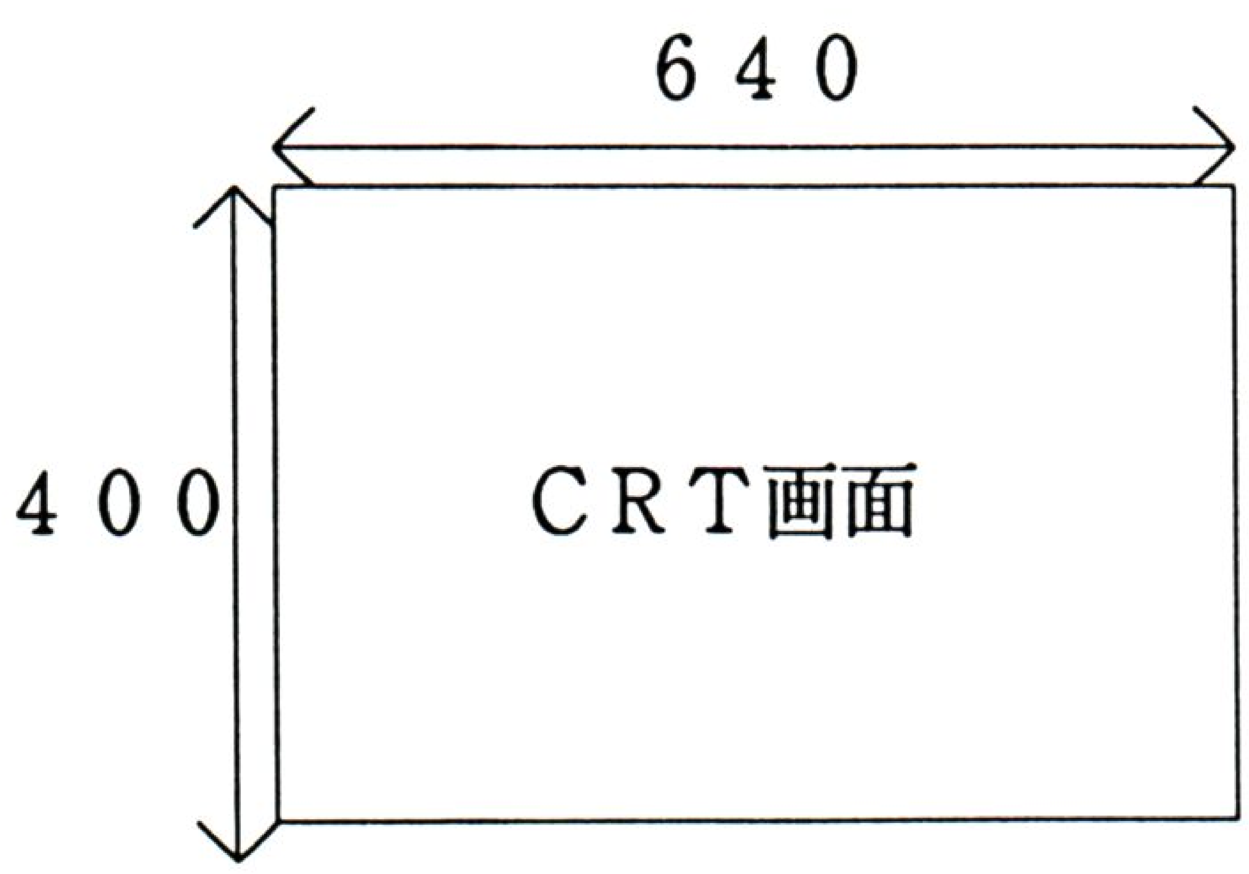
RANDOMIZEは乱数系列を取り替えるステートメントで、TIMERは、今日の０時からの秒数なのです。例えばRANDOMIZE 12345とすると、やはりいつも同じ乱数系列が使われてしまいます。そこで、毎秒ごと時刻が変わるTIMERを使ったのです。次のプログラムで秒数が変わるようすを見てください。

```
'QB2_18_3
FOR I =1 TO 10000
    PRINT TIMER
NEXT I
```

さていよいよ、箱を描くプログラムをつくります。

```
'QB2_18_4
RANDOMIZE TIMER
FOR I = 1 TO 1000
 LINE (RND * 640, RND * 400)-(RND * 640, RND * 400), RND * 1 , B
 LINE (RND * 640, RND * 400)-(RND * 640, RND * 400), RND * 1 , BF
NEXT I
```

640を乗じたのは画面が横方向に640の点からなっているからで、400を乗じたのは画面が縦方向に400の点からなっているからです。

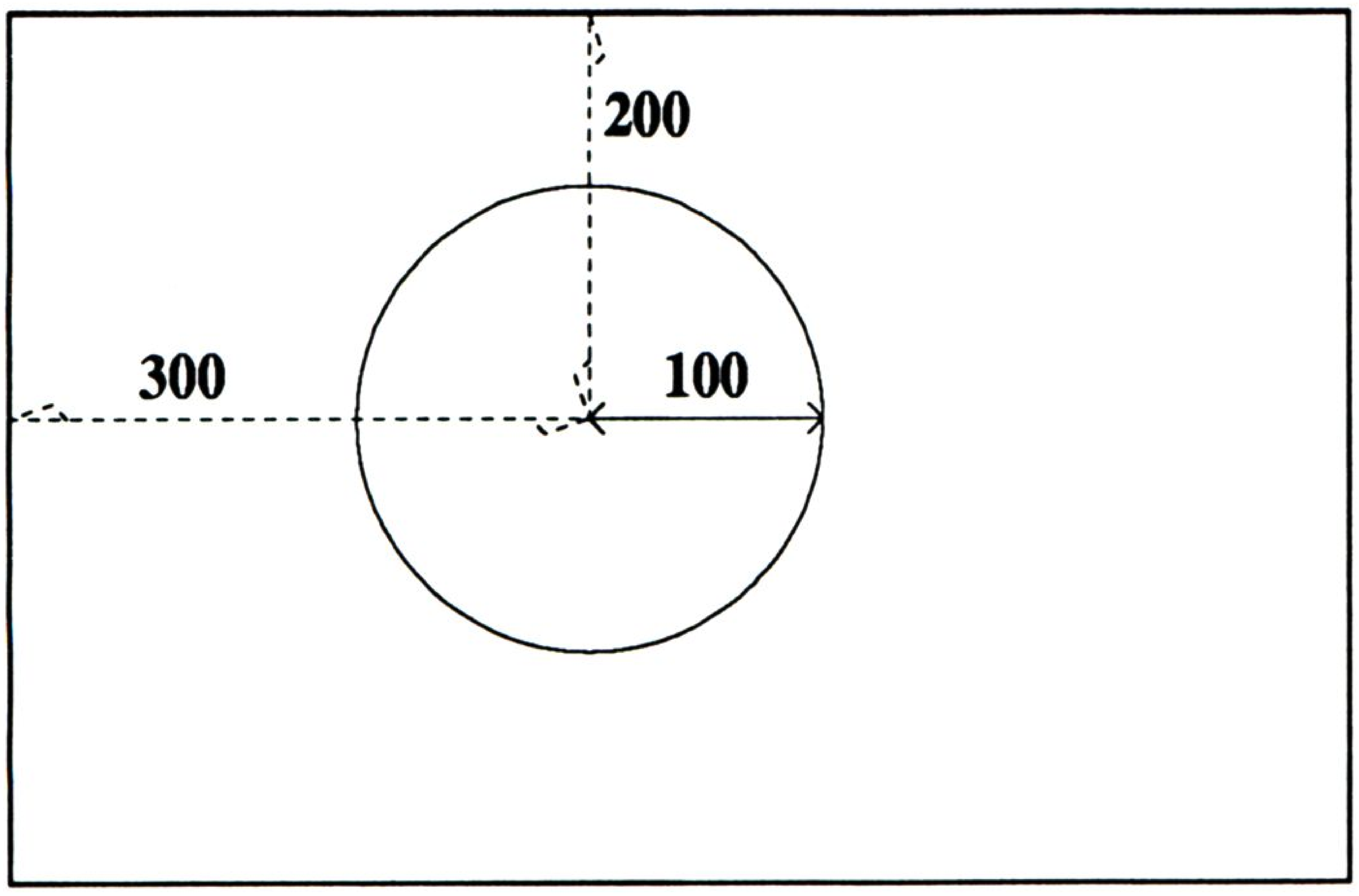


２．１９　円の模様

次のプログラムは円を描きます。

```
'QB2_19_1
CLS
CIRCLE (300, 200), 100, 3
```

(300,200)は円の中心の座標、100は半径、3は色番号（水色）です。



CIRCLE (中心の位置), 半径, 色番号

FOR....NEXTを使って、次のようにいろいろな円模様をつくることができます。

```
'QB2_19_2
CLS
 FOR I = 0 TO 350 STEP 5
     CIRCLE (300, 200), I, I
 NEXT I
```

```
'QB2_19_3
CLS
FOR I = 0 TO 640 STEP 5
    CIRCLE (I, 200), I, I
NEXT I
```

```
'QB2_19_4
CLS
FOR I = 0 TO 640 STEP 5
    CIRCLE (I, 200), 100, I
NEXT I
```

```
'QB2_19_5
CLS
FOR I = 0 TO 640 STEP 5
    CIRCLE (I, I), I, I
NEXT I
```

```
'QB2_19_6
CLS
FOR X=0 TO 640 STEP 160
    FOR Y=0 TO 400 STEP 160
        FOR J=1 TO 30
            FOR I=1 TO 80 STEP J
                CIRCLE (X, Y), I, J
            NEXT I
        NEXT J
    NEXT Y
NEXT X
```

上の例を参考にして、円の模様をいろいろ工夫してみてください。

2．20　大円小円に色を塗る

まず、乱数を使って画面いっぱいにたくさんの円を描きます。

```
'QB2_20_1
RANDOMIZE TIMER
CLS
FOR I = 1 TO 100
    CIRCLE (RND * 640, RND * 400), RND * 50, RND * 8
NEXT I
```

この円に色を塗ることにします。色は、PAINTステートメントを使って塗ります。PAINTステートメントでは、塗る位置、塗る色、塗る領域を囲む縁の色を指定する必要があります。

そこで、それぞれのデータをあらかじめ変数に入れておいて使うことにします。

```
'QB2_20_2
RANDOMIZE TIMER
CLS
FOR I = 1 TO 100
    X = RND * 640              '円の中心のX座標
    Y = RND * 400              '円の中心のY座標
    R = RND * 50               '半径
    C = RND * 8                '色
    CIRCLE (X, Y), R, C               '円
    PAINT (X, Y), RND * 8, C          '色塗
NEXT I
```

X = RND * 640 は、円の中心のX座標です。一様乱数のRNDが0から1までの範囲の数値を返すので、640を乗ずると、Xは、0から640までの数値になります。また、Y = RND * 400は、円の中心のY座標です。一様乱数に400を乗じて、Yは、0から400までの数値になります。

R = RND * 50 は、半径です。半径が50ピクセル（画面の光る1点が1ピクセル）以内になるようにしました。

RND * 8 は、色番号です。色番号は、0から7までですから、一様乱数を乗じて、8未満の数値を得ています。

PAINTステートメントの使い方は次のとおりです。

PAINT (塗る位置), 塗る色番号, 塗る領域を囲む色番号

次の例を参考にして円模様のプログラムをつくりましょう。

```
'QB2_20_3
CLS
FOR I = 350 TO 0 STEP -10
    CIRCLE (300, 200), I, I
    PAINT (300, 200), I, I
NEXT I
```

```
'QB2_20_4
CLS
FOR I = 640 TO 0 STEP -5
    CIRCLE (I, 200), I, I
    PAINT (I, 200), I, I
NEXT I
```

```
'QB2_20_5
CLS
FOR I = 640 TO 0 STEP -5
    CIRCLE (I, 200), 100, I
    PAINT (I, 200), 100, I
NEXT I
```

```
'QB2_20_6
CLS
FOR I = 640 TO 0 STEP -5
    CIRCLE (I, I), I, I
    PAINT (I, I), I, I
NEXT I
```

２．２１　整数用の小さい容器

ここでは、具体的にイメージしやすいように変数を容器と言い替えてみました。

これまでは、処理の内容によって変数を使い分けることはしなかったのですが、実は変数にもいろいろあるのです。

```
'QB2_21_1
A = 1
FOR I = 1 TO 30
    A = A * 2
    PRINT A;
NEXT I
```

このプログラムは、最初にAに1を入れておいて、Aを30回２倍にしています。その結果は次のようになります。２倍になっていることを確かめてください。

```
2  4  8  16  32  64  128  256  512  1024  2048  4096  8192
16384  32768  65536  131072  262144  524288  1048576  2097152
4194304  8388608  1.677722E+07  3.355443E+07  6.710886E+07
1.342177E+08  2.684355E+08  5.368709E+08  1.073742E+09
```

途中からおかしな数値が出てきます。8388608の次の1.677722E+07は、1.677722×10^7という意味で16777220のことなので、ここからは正確な２倍になっていません。この変数は単精度浮動小数点型の変数といい、容器の大きさが６桁程度の精度がでるようにつくられているので、数値が大きくなると末尾が切れてしまうのです。

これよりも小さい範囲でしかも整数だけを扱う整数型の変数があります。これを使うときは、変数名に%を付けるか、「DEFINT 変数名」というように変数の型を定義して、整数型の変数であることを示さなければなりません。

次のプログラムは、両方の方法で1の２倍を30回繰り返すプログラムです。

```
'QB2_21_2
A% = 1
FOR I% = 1 TO 30
A% = A% * 2
PRINT A%;
NEXT I%
```

```
'QB2_21_3
DEFINT A,I
A = 1
FOR I = 1 TO 30
A = A * 2
PRINT A;
NEXT I
```

どちらのプログラムでも、次のように同じ結果になります。16384までは正確に計算されますが、その次がオーバーフローということで停止してしまいます。文字どおり変数の容器がいっぱいになったのです。整数型の変数は、-32,768から32,767までの数値を扱うことのできる変数なのです。

```
2  4  8  16  32  64  128  256  512  1024  2048  4096  8192
16384 (オーバーフローのため停止)
```

整数型の変数を利用するメリットは、容器が小さいので、高速に計算や読み書きができるというところにあります。これまで扱ってきたプログラムは、すべてが整数型の変数で十分間に合います。

DEFINTは、DEFINE（定義）とINTEGER（整数）を合成した言葉です。DEFINT I-Kとすると、I、J、K、IWATE、JAPAN、KAISYAなど、I、J、Kを頭文字とする容器がすべて整数型の変数として取り扱われます。

実際には、次のような使い方をします。

```
'QB2_21_4
DEFINT I-J, X-Y
FOR X = 0 TO 640 STEP 160
    FOR Y = 0 TO 400 STEP 160
        FOR J = 1 TO 30
            FOR I = 1 TO 80 STEP J
                CIRCLE (X, Y), I, J
            NEXT I
        NEXT J
    NEXT Y
NEXT X
```

２．２２　整数用の大きい容器

整数型の変数では、大きな整数を取り扱うことができないのですが、特に大きい整数のための変数が用意されております。これを倍長整数型の変数といいます。

次のプログラムは、倍長整数型の変数の使用例です。

```
'QB2_22_1
A& = 1
FOR I = 1 TO 30
      A& = A& * 2
      PRINT A&;
NEXT I
```

```
'QB2_22_2
DEFLNG A
A = 1
FOR I = 1 TO 30
      A = A * 2
      PRINT A;
NEXT I
```

このプログラムは同じように動作し、２倍の計算を繰り返して2の1乗から30乗までを表示します。変数名に&を付けるか、または「DEFLNG 変数名」と定義すれば、倍長整数として扱われます。DEFは、DEFINE（定義）の略で、LNGはLONGの略です。また、DEFLNG A-Cとすると、A、B、Cを頭文字とする変数は倍長整数型の変数となります。

```
 2  4  8  16  32  64  128  256  512  1024  2048  4096  8192
16384  32768  65536  131072  262144  524288  1048576  2097152
4194304  8388608  16777216  33554432  67108864  134217728
268435456  536870912  1073741824
```

ここでFOR I = 1 TO 30 の30を31に変更して実行すると、３１回目にオーバーフローとなって停止します。倍長整数型の変数は、-2,147,483,648から2,147,483,647までの数値を取り扱うことができます。これを1だけオーバーしたのです。

２．２３　小数点のある数値の容器

整数は小数点のない数値ですが、小数点のある数値のための容器は別にあります。すでに２．２１で説明した単精度浮動小数点型の変数がそれです。実は、プログラムの中で特に定義をしないで使用した変数はすべて単精度浮動小数点型の変数として取り扱われますから、知らずに使用してきたのです。

この単精度浮動小数点型の変数がどれほどの数値を扱うことができるかをテストするプログラムをつくります。

```
'QB2_23_1
DEFINT I
A = 1
FOR I = 1 TO 127
    A = A * 2
    PRINT "I="; I,"A="; A
NEXT I
```

```
I= 1          A= 2
I= 2          A= 4
I= 3          A= 8
I= 4          A= 16
I= 5          A= 32
      (省略)
I= 124        A= 2.126765E+37
I= 125        A= 4.25353E+37
I= 126        A= 8.507059E+37
I= 127        A= 1.701412E+38
```

'QB2_23_1のプログラムは、1から始めて結果の２倍を１２７回繰り返すものです。

１２７回目の答は1.701412E+38とありますが、これは、1.701412×10^{38}ということです。大きな数を表しているのですが表示されている数の桁数は全部で７桁だけです。最下位の数はいろいろ処理されて誤差が生じているので、信頼できるのは上位からおよそ６桁ぐらいです。

コンピュータは、このように指定された変数型に必要とする桁数の精度で計算しているのです。

単精度浮動小数点型の変数は、-3.402823×10^{38}から-1.175494×10^{-38}まで、及び$+1.175494\times10^{-38}$から$+3.402823\times10^{38}$までを取り扱うことができます。

このプログラムでFOR I = 1 TO 128 に変更して実行すると１２８回目にオーバーフローのため停止します。

単精度浮動小数点型の変数は、A!,B!,C!,D!のように!を付して使用する方法と、「DEFSNG 変数名」と定義して使用する方法があります。SNGはSINGLEの略です。

何もしなければ単精度浮動小数点型ですから、一般にはDEFSNGや!を使わないようです。

さらに、もっと精度を高めるために、倍精度浮動小数点型の変数があります。

倍精度浮動小数点型の変数は、DEFDBL A-D と定義したり、個別に、A#、B#、C#、D#というように#を付して定義して使用します。

有効桁数はおよそ15桁で、10の308乗という気の遠くなりそうな大きな数を扱うことができます。

```
'QB2_23_2
DEFINT I
DEFDBL A
A = 1
FOR I = 1 TO 1023
    A = A * 2
    PRINT "I="; I,"A="; A
NEXT I
```

```
I= 1          A= 2
I= 2          A= 4
I= 3          A= 8
I= 4          A= 16
     (省略)
I= 1020       A= 1.123558209288947D+307
I= 1021       A= 2.247116418577895D+307
I= 1022       A= 4.49423283715579D+307
I= 1023       A= 8.98846567431158D+307
```

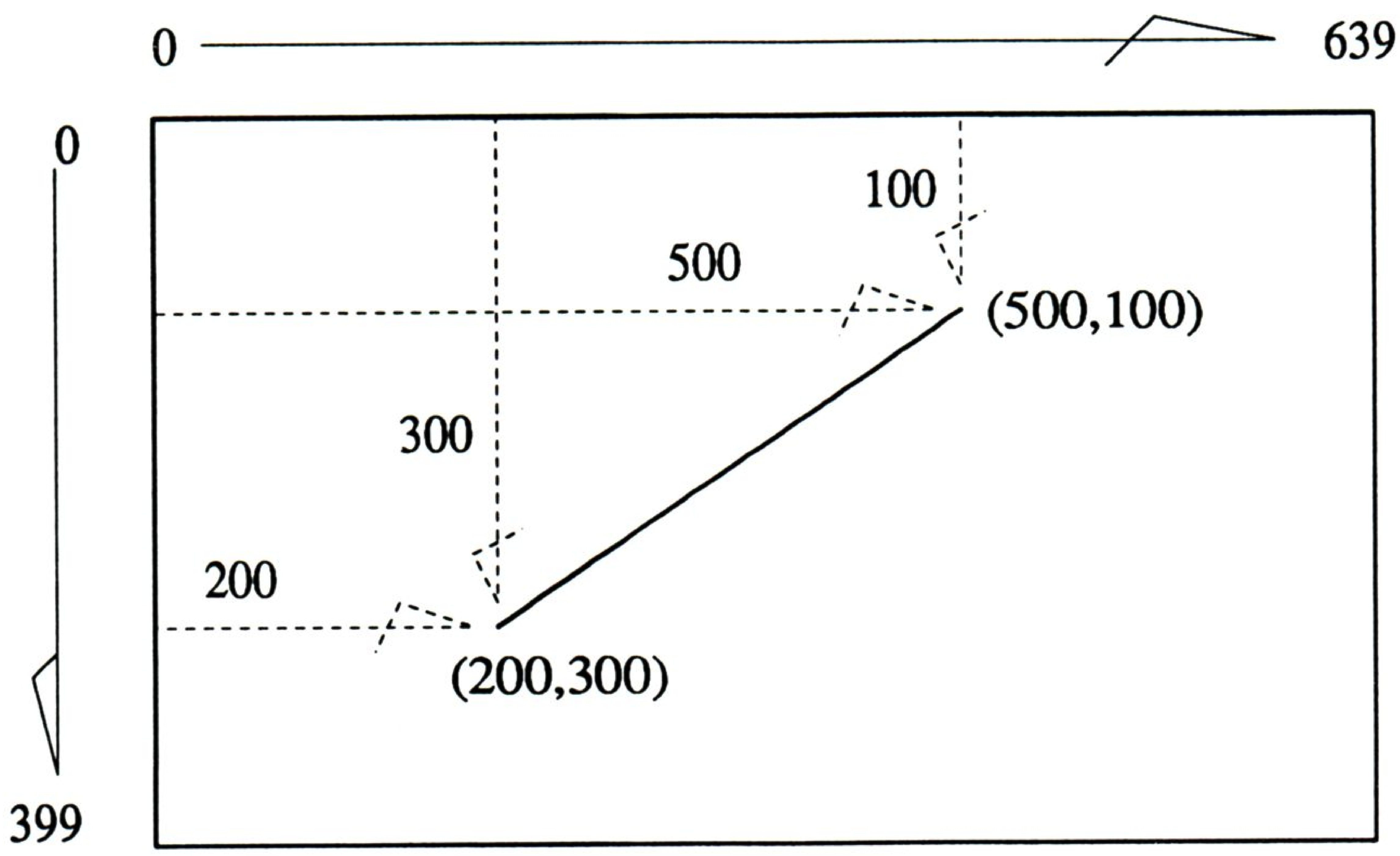
他に、文字専門の容器もあるのですが、この説明はあとにします。

２．２４　線を描く

線は、箱の描き方と同じです。LINEステートメントはもともと線を描く命令で、これにBを加えて長方形を描くようにしたものなのです。

```
'QB2_24_1
CLS
LINE (200, 300)-(500, 100), 4
```

(200, 300)-(500, 100)は線の両端の座標で、4は色番号（赤）です。



次のプログラムは、乱数を使っていろいろな線を描いた例です。

```
'QB2_24_2
DEFINT I
CLS
FOR I = 1 TO 200
    LINE (RND * 640, RND * 400)-(RND * 640, RND * 400), RND * 100
NEXT I
```

次のプログラムは、相対座標を使って連続した線を描いた例です。

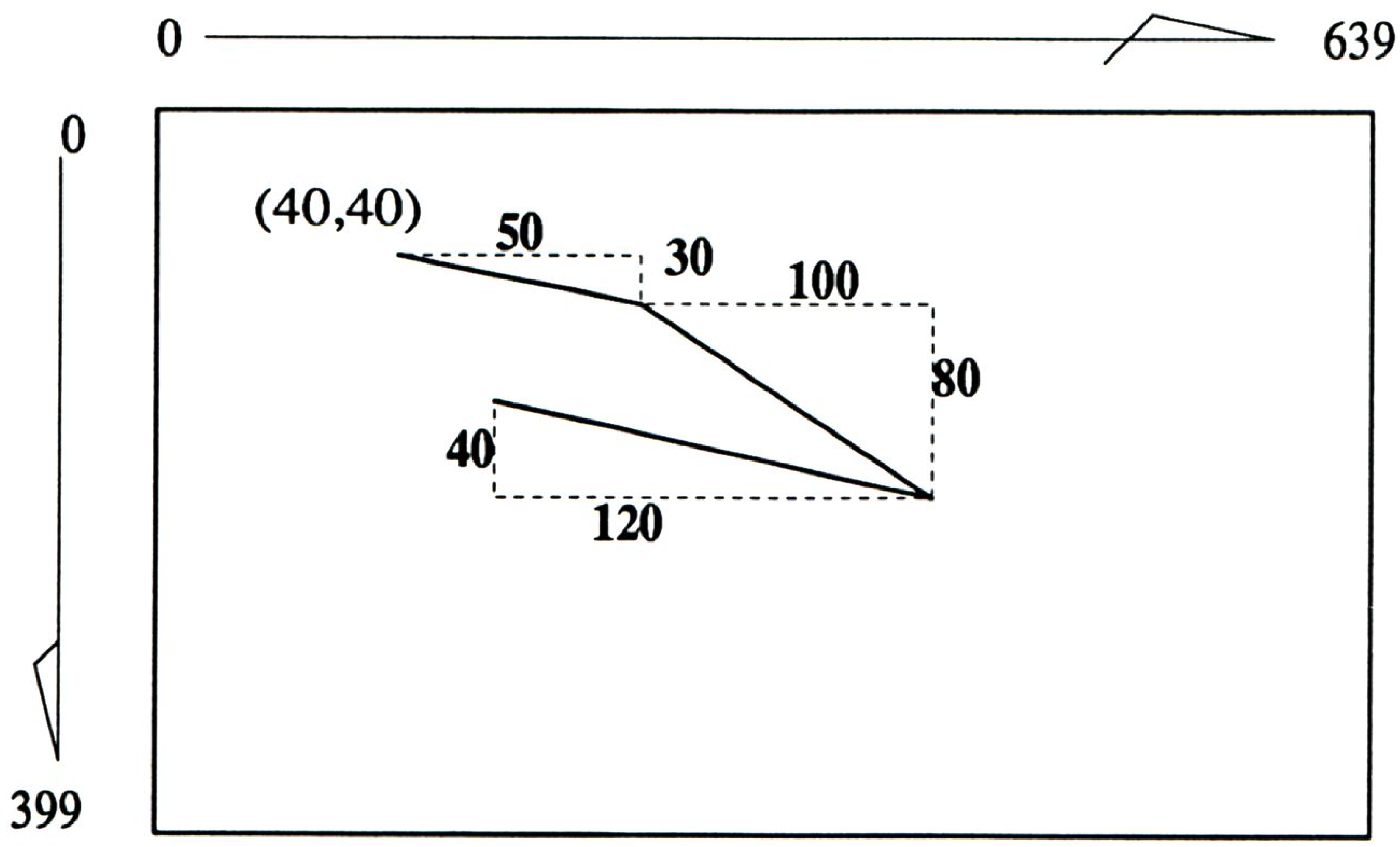
```
'QB2_24_3
CLS
LINE (40, 40)- STEP (50, 30), 3
LINE- STEP (100, 80), 4
LINE- STEP (-120, -40), 5
```

STEPを使うと、現在位置からの相対的な座標で線を描くことができます。

LINE (40, 40)- STEP (50, 30), 3 は、(40,40)の位置から右へ50、下へ30の位置へ線を描きます。

LINE- STEP (100, 80), 4は、現在位置から右へ100、下へ80の位置へ線を描きます。

LINE- STEP (-120, -40), 5は、現在位置から左へ120、上へ40の位置へ線を描きます。



2．25　線で縞模様

線で縞模様を描きます。
次のプログラム例を参考にして、線の縞模様のプログラムを工夫しましょう。

次のプログラムは横縞の例です。

```
'QB2_25_1
DEFINT I
CLS
FOR I = 0 TO 399 STEP 10
    LINE (0, I)-(639, I), I
NEXT I
```

次のプログラムはたて縞の例です。

```
'QB2_25_2
DEFINT I
CLS
FOR I = 0 TO 639 STEP 10
    LINE (I, 0)-(I, 399), I
NEXT I
```

次のプログラムは、STEPを変更して何度も上描きして色を塗る例です。

```
'QB2_25_3
DEFINT I-J
CLS
FOR J = 1 TO 80
    FOR I = 0 TO 399 STEP J
        LINE (0, I)-(639, I), J
    NEXT I
NEXT J
```

次のプログラムは、階段のような模様の例です。

```
'QB2_25_4
DEFINT C,I-K
CLS
FOR J = 1 TO 30                              'STEPのキザミを変える
    FOR I = 1 TO 640 STEP J                  '階段の広さと高さを変える
        C = RND * I                          '色を変える
        LINE (0, 0)- STEP (I, 0), C             'つくりはじめの階段
            FOR K = 1 TO 10                     '階段は１０段ずつつくる
                LINE - STEP (0, I), C              '階段の縦部分
                LINE - STEP (I, 0), C              '階段の横部分
            NEXT K
    NEXT I
NEXT J
```

2．２６　点線をつくる

点線を描くプログラム例は、次のとおりです。

```
'QB2_26_1
CLS
LINE (100, 200)-(400, 300), 5, , &HF0F0
```

実行結果

```
'QB2_26_2
CLS
LINE (100, 200)-(400, 300), 5, , &HFF00
```

実行結果

点線は、１６個の光の点（16ピクセル）を&HF0F0という形に表現して、これを画面上に表示するようになっています。

最初の&Hは、次のF0F0が１６進数であることを示しています。

FF00は点線の形状を表しています。一つの１６進数が４個の光る点を受けもっており、１６進数のFは２進数では1111でこれで４個の点を光らせ、１６進数の0は２進数でも0で４個の点は光らせないということです。

画面上の１文字分の巾

点線の形状	――――								――――							
画面上の点	光	光	光	光	黒	黒	黒	黒	光	光	光	光	黒	黒	黒	黒
２進数で表す	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
１６進数で表す	F				0				F				0			

↓

LINEステートメント上での表し方　　&HF0F0

１６進数というところで、突然Fが出てきました。Fは数字なのでしょうか。

普通に使っている0から9までの１０種類の数字では１６進数を表現できないので、アルファベットから借りて、0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、Fの１６種類の数字で表現しているのです。

参考までに、１０進数、２進数、１６進数の対応表を示します。

１０進数	２進数	１６進数
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9

１０進数	２進数	１６進数
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
18	10010	12
19	10011	13

次のプログラムは、破線の例です。

```
'QB2_26_3
CLS
LINE (100, 200)-(400, 300), 5, ,&HFFBC
```

Bを加えると、長方形も点線で描けます。

```
'QB2_26_4
CLS
LINE (100, 200)-(400, 300), 5, B,&HFFBC
```

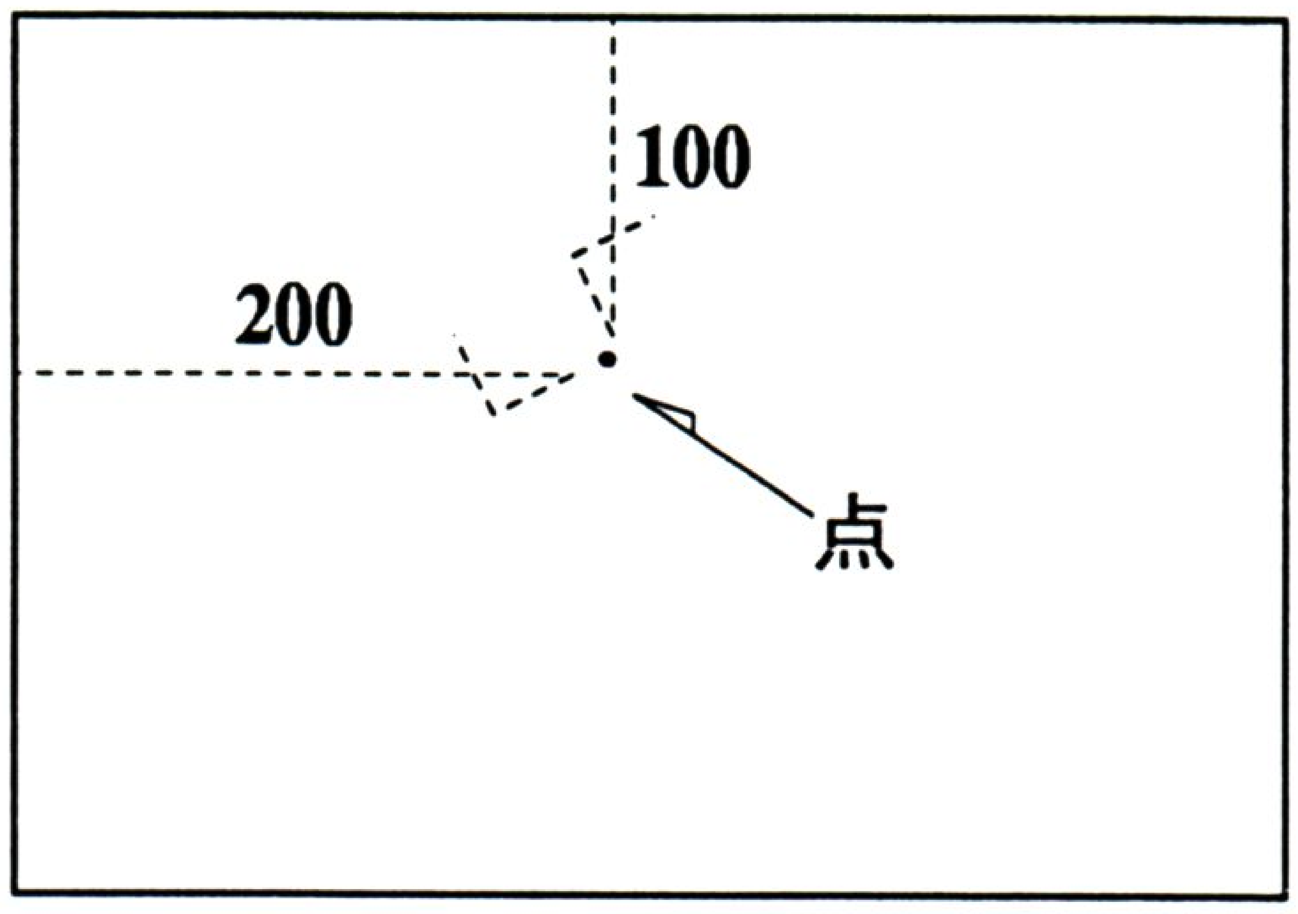
2．27　星空と流れ星

パソコンの画面上に星空を描きましょう。

１個の星は点で表し、次のようにプログラムを書きます。

```
'QB2_27_1
CLS
PSET (200, 100), 7

 (この点を消す場合は)
PRESET (200, 100)
```



点の描き方と消し方は、次のとおりです。

点を描く	PSET (点の位置), 色番号
点を消す	PRESET (点の位置)

そこで、乱数を使って星空を描きます。

```
'QB2_27_2
RANDOMIZE TIMER                    '乱数系列を設定
DEFINT I                           'I は整数
CLS                                ' 画面消去
FOR I = 1 TO 300                   '300回繰り返す
    PSET (RND * 640, RND * 400), I MOD 8    '点を描く
NEXT I
```

色番号の部分は、I MOD 8 と書いています。

色番号は0から7までですが、これ以外の数でも一応色指定ができるので、これまでは間に合わせてきました。ここでは、I MOD 8 として、かならず0から7までの整数になるようにしました。

これは、Iを8で割って余りを計算する式です。この式の答は、割り切れる場合は0で、割り切れない場合は1、2、3、4、5、6、7となりますから、必ずいずれかの色番号と一致することになります。

次に、流れ星を加えたプログラムをつくります。

```
'QB2_27_3
RANDOMIZE TIMER
DEFINT I-J
CLS
FOR I = 1 TO 300
    PSET (RND * 640, RND * 400), I MOD 8
NEXT I
FOR I = 1 TO 10                         '流れ星を10個表示する
    X1 = RND * 640                      '流れ星の始点の横方向の位置
    Y1 = RND * 400                      '流れ星の始点の縦方向の位置
    X2 = RND * 640                      '流れ星の終点の横方向の位置
    Y2 = RND * 400                      '流れ星の終点の縦方向の位置
    LINE (X1, Y1)-(X2, Y2), 7           '流れ星を描く
    LINE (X1, Y1)-(X2, Y2), 0           '流れ星を消す
    FOR J = 1 TO RND * 10000
                                '次の流れ星を表示するまでの時間をとる
    NEXT J
NEXT I
```

RND * 10000 は、流れ星を表示する時間間隔を変えるために、乱数に10000を乗算して、それを数えて、時間の間隔をとるようにしたのです。

２．２８　方程式の計算と見やすい数字の表示

数学の式を計算しましょう。

$Y=X^3-3X-1$ のプログラムをつくります。

```
'QB2_28_1
CLS
FOR X = -3 TO 3 STEP .3
    Y = X ^ 3 - 3 * X - 1
    PRINT "X="; X, "Y="; Y
NEXT X
```

```
X=-3            Y=-19
X=-2.7          Y=-12.583
X=-2.4          Y=-7.624001
X=-2.1          Y=-3.961002
X=-1.8          Y=-1.432001
X=-1.5          Y= .1249993
X=-1.2          Y= .8719996
X=-.9000003     Y= .9710002
X=-.6000003     Y= .5840005
X=-.3000003     Y=-.1269993
X=-2.384186E-07 Y=-.9999993
X= .2999998     Y=-1.872999
X= .5999998     Y=-2.584
X= .8999998     Y=-2.971
X= 1.2          Y=-2.872
X= 1.5          Y=-2.125001
X= 1.8          Y=-.5680017
X= 2.1          Y= 1.960996
X= 2.4          Y= 5.623995
X= 2.7          Y= 10.58299
X= 3            Y= 16.99999
```

FOR X = -3 TO 3 STEP .3は、X=-3からX=3まで、0.3きざみで計算を繰り返そうとしています。

X ^ 3 は、Xの3乗ということです。 ^は帽子のようなのでハットといいます。Xの8乗ならば、X ^ 8 とするのです。

右の計算結果は、一見詳細な結果が出ているようですが、よく見ると筆算の計算結果と違っているところがたくさんあります。例えば、-0.9となるべきところが、-0.9000003となっています。これは、X、Yが単精度浮動小数点型の変数なので、誤差が出てしまうのです。

次に、計算結果が見やすく表示されるプログラムをつくります。

```
'QB2_28_2
CLS
FOR X = -3 TO 3 STEP .3
      Y = X ^ 3 - 3 * X - 1
      PRINT USING "X=###.# Y=###.###";X;Y
NEXT X
```

```
X= -3.0  Y=-19.000
X= -2.7  Y=-12.583
X= -2.4  Y= -7.624
X= -2.1  Y= -3.961
X= -1.8  Y= -1.432
X= -1.5  Y=  0.125
X= -1.2  Y=  0.872
X= -0.9  Y=  0.971
X= -0.6  Y=  0.584
X= -0.3  Y= -0.127
X= -0.0  Y= -1.000
X=  0.3  Y= -1.873
X=  0.6  Y= -2.584
X=  0.9  Y= -2.971
X=  1.2  Y= -2.872
X=  1.5  Y= -2.125
X=  1.8  Y= -0.568
X=  2.1  Y=  1.961
X=  2.4  Y=  5.624
X=  2.7  Y= 10.583
X=  3.0  Y= 17.000
```

PRINT USING "X=###.#　Y=###.#";X;Y の中の USING "###.#";は、XとYの表示の形を定めています。###は整数部３桁ということで、次の.#は小数部１桁ということです。例えば、整数部４桁で小数部２桁ならば、USING "####.##"と書きます。

また、整数部の桁数が少ない場合は先頭にスペースが入り、端数がある場合は自動的に四捨五入をして表示されます。例えば-3.961002を"###.#"で表示すると、先頭にスペースが１個入り、次に小数第２位が四捨五入されて-4.0と表示されます。

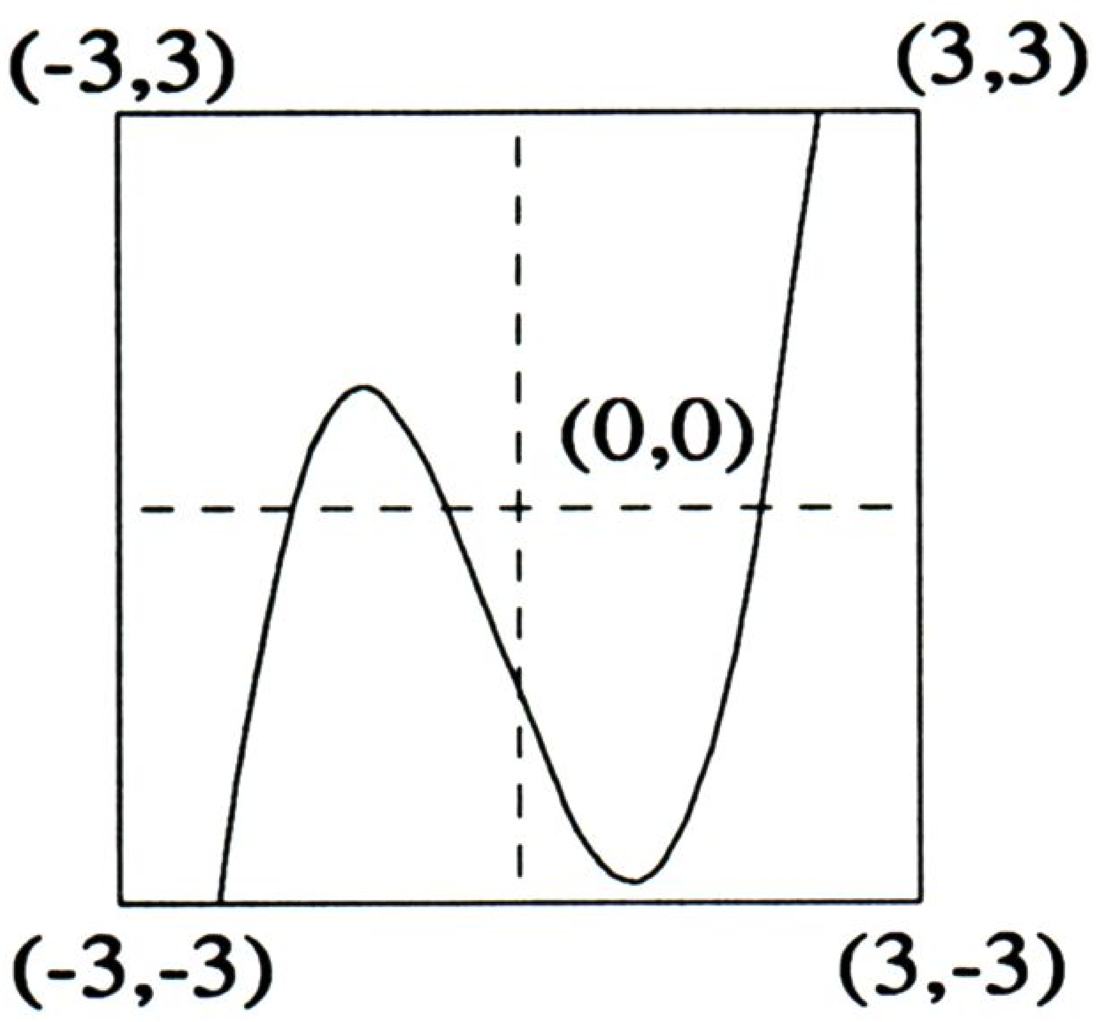
2．29　方程式を描く

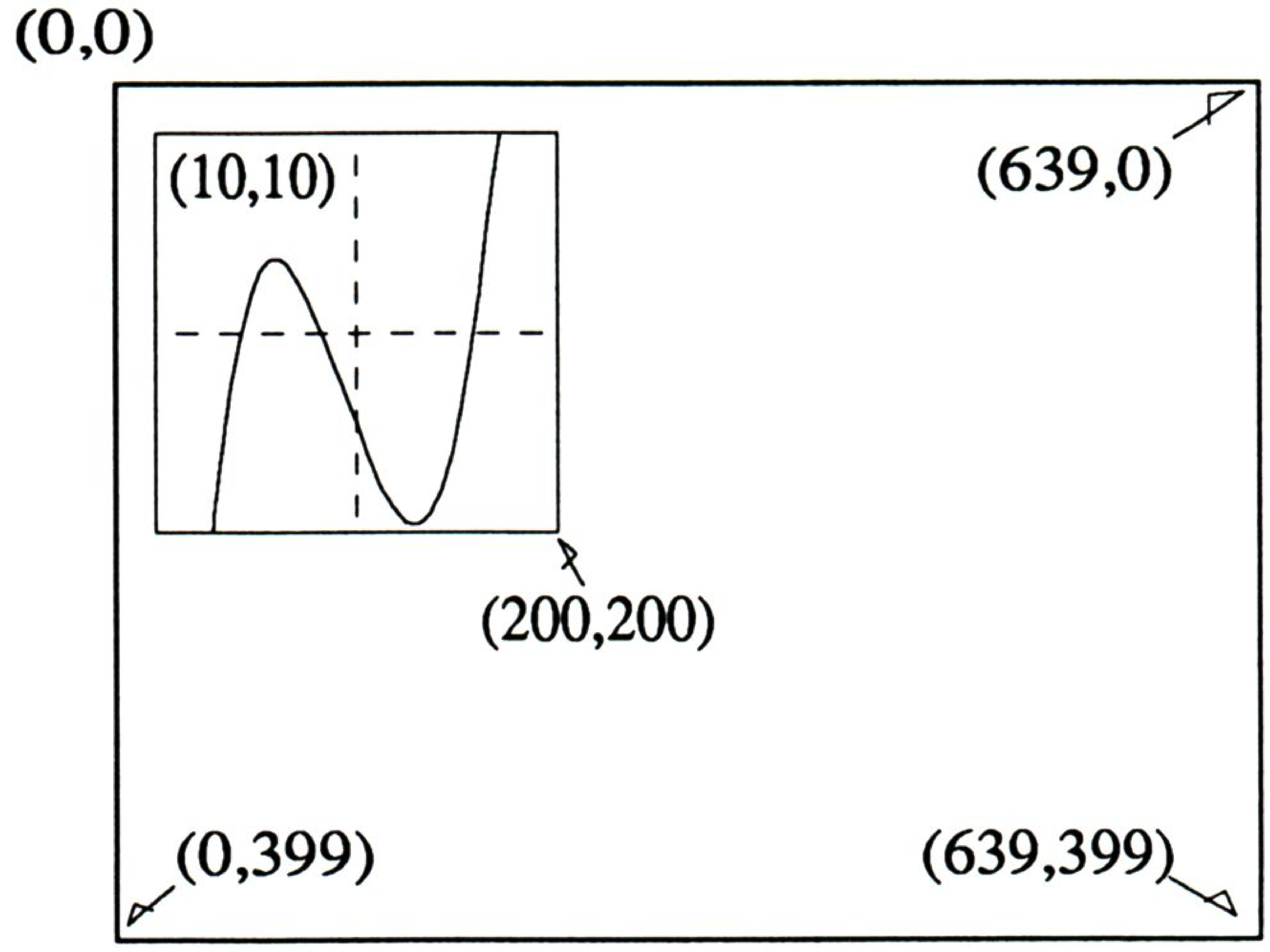
方程式のグラフを画面上に表示するプログラムをつくります。

式は、$Y=X^3-3X-1$ で、表示の範囲は、(-3,-3) から (3,3) までとします。

ここで、これから描こうとする方程式は左下が (-3,-3) で、右上が (3,3) の座標なのに、パソコンの画面は左上が (0,0) で右下が (640,400) ですから、まったく違っているのです。

Ｑｕｉｃｋ　ＢＡＳＩＣには便利な道具があって、(-3,-3) から (3,3) までの仮の座標にグラフを描くと、実際の画面に表示してくれます。





(-3,-3) から (3,3) までの範囲にグラフを描いて、実際のスクリーン上の、(10,10) から(200,200)までの位置に表示するプログラムをつくります。

```
'QB2_29_1
CLS
WINDOW (-3, -3)-(3, 3)          '(-3, -3)から(3, 3)まで窓を用意して
VIEW (10, 10)-(200, 200)        '窓を(10, 10)から(200, 200)までに表示
LINE (-3, -3)-(3, 3), 3, B, &HFFFF    '表示する部分の枠を描く
LINE (-3, 0)-(3, 0), 3, , &HF0F0      'X軸を描く
LINE (0, -3)-(0, 3), 3, , &HF0F0      'Y軸を描く
FOR X = -3 TO 3 STEP .01              'X=-3から3まで0.01きざみ
                                      'で繰り返す
    Y = X ^ 3 - 3 * X - 1             'Yを計算
    PSET (X, Y), 2                    '点を描く
NEXT X
```

次は、Y=X^3-5X^2+3X のグラフを(-2,15)から(6,-10)の範囲に描いて、画面上は(250,210)から(450,390)の範囲に表示するプログラムです。

```
'QB2_29_2
CLS
WINDOW (-2, 15)-(6, -10)               '(-2, 15)から(6, -10)までの範囲を見る
VIEW (250, 210)-(450, 390)             'それを(250, 210)から(450, 390)
                                       'までに表示する
LINE (-2, 15)-(6, -10), 6, B,&HFFFF    '表示する部分の枠を表示する
LINE (-2, 0)-(6, 0), 6, ,&HF0F0        'X軸を描く
LINE (0, 15)-(0, -10), 6, ,&HF0F0      'Y軸を描く
FOR X = -2 TO 6 STEP .01               'X=-2から6まで0.01きざみで繰り返す
    Y = X ^ 3 - 5 * X * X + 3 * X      'Yを計算
    PSET (X, Y), 5                     '点を描く
NEXT X
```

2．30　円弧と扇の模様

円弧は、円と同様にCIRCLEステートメントで容易に描くことができます。

次のプログラムは、半径100、30度から120度までの円弧を緑色で描きます。

```
'QB2_30_1
CLS
CIRCLE (300,200), 100, 2, 30 * 3.14 / 180, 120 * 3.14 / 180
```

円弧は次のように描きます。

CIRCLE (円の中心の位置) , 半径, 色番号, 始めの角度, 終りの角度

始めの角度と終りの角度は弧度法による角度ですから、度数法による30度と120度は、弧度になおす必要があります。

弧度法の角度＝度数法の角度×3.14÷180 です。

例えば度数法30度ならば、30 * 3.14 / 180 と計算します。

円弧の連続模様のプログラム例です。円弧の模様は、単純に並べるだけでもおもしろい連続模様が描けます。

```
'QB2_30_2      円弧の連続模様
RANDOMIZE TIMER
CLS
FOR X = 0 TO 640 STEP 40
    FOR Y = 40 TO 400 STEP 40
        CIRCLE (X, Y), 40, (X * Y + 1) * RND MOD 7 + 1,
                                    0, 90 * 3.14 / 180
    NEXT Y
NEXT X
```

半径40で0度から90度までの円弧を、画面全体に表示するため、XとYで横位置と縦位置とを決めています。

色番号は、(X * Y + 1) * RND MOD 7 + 1　と乱数を使って難しくしていますが、いろいろに取り替えて好みの色にしてください。90 * 3.14 / 180は、90度の弧度の計算ですが、90を360度以内で変えると模様も変わります。半径40ですが、これもいろいろに変えることができます。

次は、扇を描きます。扇は円弧の開始角度と終了角度をマイナスとすれば描けます。

次のプログラムは、QB2_30_1のプログラムを扇に改造したものです。

```
'QB2_30_3 半径100、30度から120度までの扇を緑色で描く
CLS
CIRCLE (300,200), 100, 2, -30 * 3.14 / 180, -120 * 3.14 / 180
```

次のプログラムは、いろいろな形の扇を画面全体に表示します。

```
'QB2_30_4
RANDOMIZE TIMER
CLS
FOR I = 1 TO 200
    CIRCLE (RND * 640, RND * 400), RND * 100, I MOD 7 + 1,
                        -RND * 2 * 3.14, -RND * 2 * 3.14
NEXT I
```

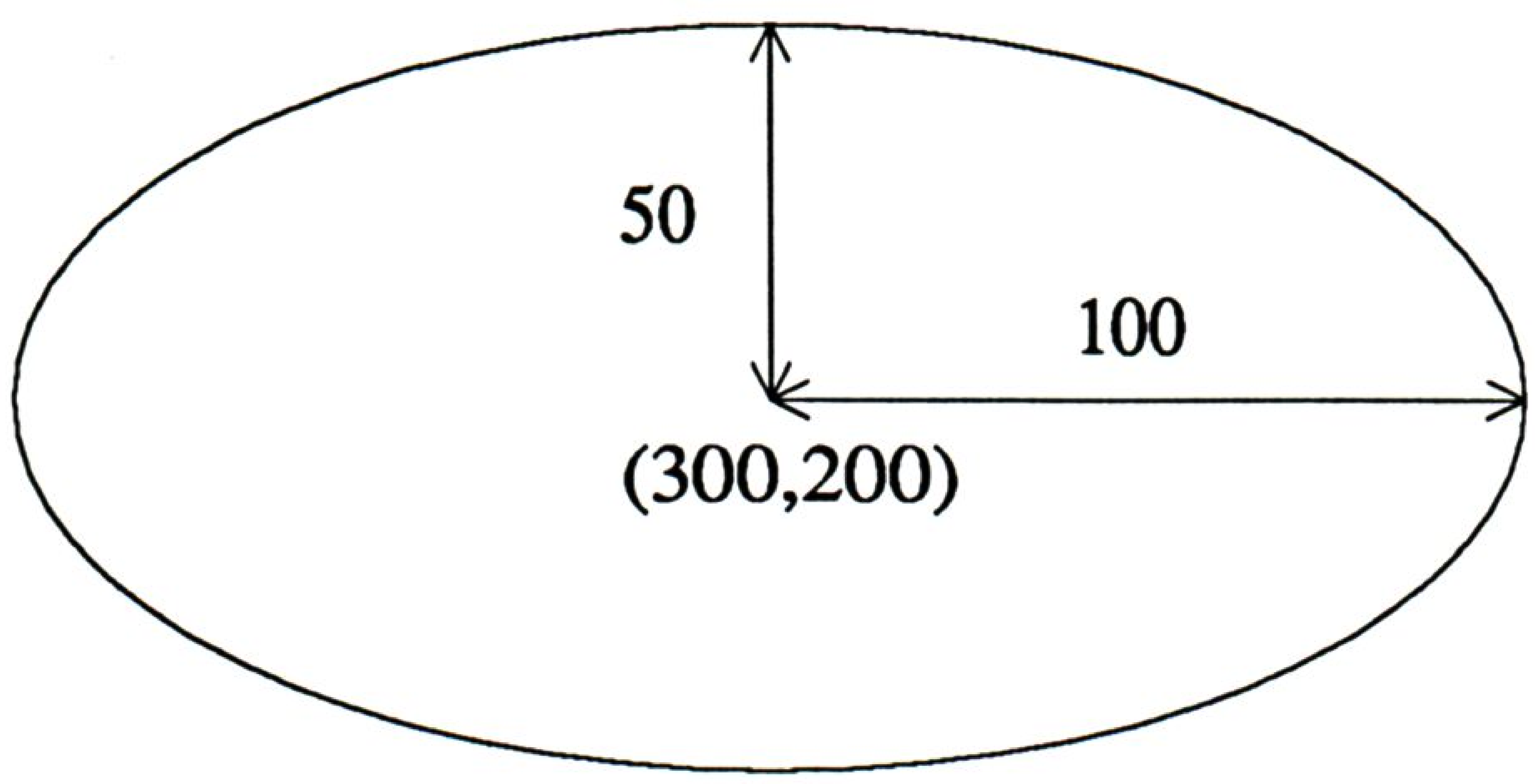
2．３１　楕円の模様

楕円は、円のCIRCLEステートメントに縦横の径の比率を加えて描きます。
楕円のステートメントの形式は、次のとおりです。

CIRCLE (楕円の中心の座標), 楕円の横軸の半径, 色番号 ,,, 縦横の比率

次のプログラムは、横軸の半径100、縦軸の半径50の楕円を画面上の(300,200)の位置へ水色で描きます。

```
'QB2_31_1
CLS
CIRCLE (300, 200),100, 3, , , 50 / 100
```



$$比率 = \frac{50}{100} = 0.5$$

また、楕円の円弧や扇も、円の場合と同様に描くことができます。

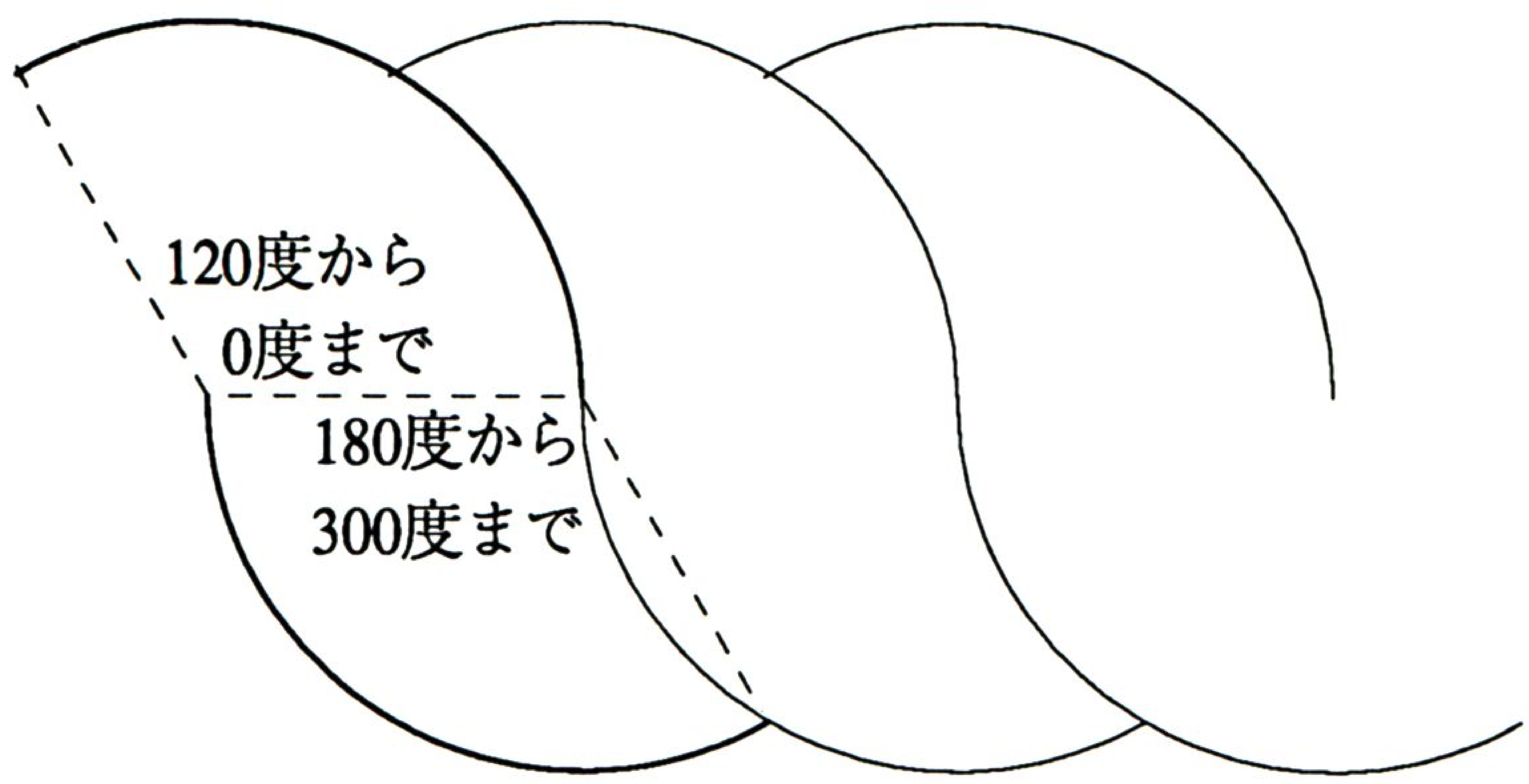
楕円の円弧　　CIRCLE (楕円の中心の位置), 楕円の横軸の半径, 色番号,
開始角度, 終了角度, 縦横の比率

楕円の扇　　CIRCLE (楕円の中心の位置), 楕円の横軸の半径, 色番号,
-開始角度, -終了角度, 縦横の比率
↑　↑
マイナスとします

楕円の連続模様を作りましょう。次のプログラムは、横軸の半径40の楕円を、比率0.2から2まで0.2きざみで変えて、画面全体に表示します。

```
'QB2_31_2
FOR R = .2 TO 2 STEP .2                     '比率を0.2から2まで0.2きざみで
                                                          '変えて繰り返す
    CLS                                     '画面消去
    FOR X = 0 TO 640 STEP 40                '画面上の横方向の中心位置の決定
        FOR Y = 0 TO 400 STEP 40 * R        '画面上の縦方向の中心位置の決定
            CIRCLE (X, Y), 40, X * Y MOD 7 + 1, , , R        '楕円を描く
        NEXT Y
    NEXT X
NEXT R
```

次に、２個の円弧の連続で、しめなわ模様を描くプログラムを作ります。



次のプログラムは、半径19の楕円を、比率0.2から2まで0.2きざみで変化させながら、しめなわ模様を描きます。

```
'QB2_31_3
R=19                                    '半径を19とする
FOR P = .2 TO 2 STEP .2                 '楕円の比率は0.2から2まで0.2きざみで
                                                            '繰り返す
    CLS                                         '画面消去
    FOR Y = R TO 400 STEP R * P * 2 + 1         '中心の画面の縦位置の決定
        FOR X = 0 TO 640 STEP R                 '中心の画面の横位置の決定
            CIRCLE (X, Y), R, Y MOD 7 + 1, 0, 120 * 3.14 / 180, P
                                                            '上の楕円
            CIRCLE (X + R, Y), R, Y MOD 7 + 1,
                        180 * 3.14 / 180, 300 * 3.14 / 180, P   '下の楕円
        NEXT X
    NEXT Y
NEXT P
```

2．32　繰り返しの名人と終了

パソコンは繰り返しの名人です。

これまでも、FOR....NEXTという便利な仕掛けで繰り返しの仕事を見事にこなしてきました。FOR....NEXTは、あらかじめ繰り返す回数が決った処理をしました。ところが、実際には繰り返す回数が分らない場合や都合によって途中で中止したい場合があります。

このような処理に使える仕掛けがDO....LOOPなのです。

次のプログラムは、箱の表示を永久に繰り返します。

```
'QB2_32_1 永久の繰り返し
DO
    CLS
    PRINT "永久に繰り返します。"
    FOR I = 1 TO 640 STEP 2
        LINE (I, I)-(I * 2, I * 2), I MOD 7 + 1, B
    NEXT I
LOOP
```

このプログラムの動作を中止するには、STOP キーを押すか、強引ですが電源を切る以外ありません。

そこで、1回模様を描くたびに続けるかどうか確認するように改造します。

```
'QB2_32_2 1回描くたびに継続を確認する
DO
    CLS
    FOR I = 1 TO 640 STEP 2
        LINE (I, I)-(I * 2, I * 2), I MOD 7 + 1, B
    NEXT I
    INPUT "模様を続けますか。はい:Y いいえ:N ---";Y$
LOOP UNTIL Y$="N"
```

このプログラムでは、Nを入力すれば中止できます。その他のキーを押したら続けるようになっています。

LOOP UNTIL Y$="N" は、Y$という変数にNが入っていれば、繰り返しを中止するということです。そこで、UNTIL の後ろに中止の条件を書けばよいのです。

次のプログラムは、何かキーを押せば、そのとき描いていた円を描き終ったところで終了し、何も押さないでいればその動作を繰り返します。

```
'QB2_32_3      終了する
DO
CLS
      FOR I = 1 TO 640 STEP 5
           CIRCLE (I, I), I, I MOD 7 + 1
      NEXT I
LOOP UNTIL INKEY$ <> ""
```

前のプログラムは、Nを押すと終了し、N以外のキーを押して繰り返すようになっていますが、それとは似ているようで異なります。 INKEY$は、押されたキーの1文字を受け取る仕掛けです。INPUTは入力されるまで待っていますが、INKEY$は押されていなければ何もしません。

""は何もないコード（NULLコード）ですから、INKEY$ <> "" は、「何かキーが押されていれば」ということになり、LOOP UNTIL INKEY$ <> ""では、「何かキーが押されるまで、繰り返す。」ということになります。言い直せば、「何か押されていれば、終了する。」ということになります。

次のプログラムは、模様を描いている途中でも、中止したい時に中止できます。

```
'QB2_32_4 模様を描いている途中で中止する
DO
     CLS
     PRINT "何かキーを押すと止まります。"
     FOR I = 1 TO 640 STEP 2
          CIRCLE (I, I), I, I MOD 7 + 1
          IF INKEY$ <> "" THEN EXIT DO
     NEXT I
LOOP
```

EXIT DOは、EXITは出口で、DOはDO....LOOPのことです。そこで、この1行は、「もしも、何かキーが押されていれば、DO..LOOPを出る」、すなわち、繰り返しは中止することになります。

２．３３　便利な選択

パソコンで仕事をする場合、アレカコレカ、と仕事を選ぶ場合が多いものです。

仕事を選択するプログラムは、次のように書きます。

```
'QB2_33_1  2種類の模様を選択して表示する
 DO
     CLS
     PRINT "模様を選択して番号を入力してください。"
     INPUT "円模様:1  箱模様:2  終了:E  "; BANGOU$
                                 'BANGOU$という容器に1か2かEを受け入れる
   CLS
      SELECT CASE BANGOU$                'BANGOU$の内容によって仕事をする
         CASE "1"                        'BANGOU$が"1"の場合の仕事＝円模様
            FOR I = 1 TO 400 STEP 5
               CIRCLE (I, I), I / 2, I MOD 7 + 1
            NEXT I
         CASE "2"                        'BANGOU$が"2"の場合の仕事＝箱模様
            FOR I = 1 TO 400 STEP 2
               LINE (I, I)-(I * 2, I * 2), I MOD 7 + 1, B
            NEXT I
         CASE "E"                        'BANGOU$が"E"の場合の仕事＝終了
             EXIT DO
         CASE ELSE                  'BANGOU$が1,2,E以外の番号の場合の仕事
            PRINT "番号違いです。"        '間違った入力をお知らせする
            BEEP: BEEP: BEEP             '警告音を３回鳴らす
      END SELECT                         '選択の仕掛の終了
   LOOP
```

選択の仕掛けは、SELECT....END SELECTで、これを使うとプログラムが鮮やかに見えます。

SELECT CASE に続いて、選択の番号を入れた変数の名前（ここではBANGOU$）を書いて、さらにCASEの後に選択の番号を書きます。すると、各CASEに書いた仕事をするのです。決められた番号以外が入力された場合は、CASE ELSEのところで処理します。

どうぞ、プログラムにCASEを書き加えて、動きを楽しんでください。

さて、番号を選択するときには、例えば ２↵ というように数字のあとで↵ を押しましたが、今度は↵ を押さないで数字の入力だけで選択できるように改造します。

```
  'QB2_33_2
DO
    CLS
    PRINT "模様を選択して番号を入力してください。"
    PRINT "円模様:1  箱模様:2  終了:E "
    DO
        BANGOU$ = INKEY$
                              '押されたキーの1文字を受け入れてBANGOU$に入れる
    LOOP UNTIL BANGOU$ <> ""   'BANGOU$に何か入力されるまで繰り返します
    SELECT CASE BANGOU$
        CASE "1"
            FOR I = 1 TO 640 STEP 5
                CIRCLE (I, I), I, I MOD 7 + 1
            NEXT I
        CASE "2"
          (以下省略)
```

2．34　選択で模様の合成

今度は、選択する番号をキーボードから入力するのではなく、乱数で自動的に模様の種類を選択して表示するプログラムをつくります。

```
'QB2_34_1
RANDOMIZE TIMER                                    '乱数系列
CLS
DO
    R = RND                                        '乱数をRへ入れる
    SELECT CASE R                                  'Rの値で処理を選択する
        CASE IS < .3                               'R<0.3ならば円模様
            FOR I = 0 TO 10
                CIRCLE (RND * 640, RND * 400), RND * 100, RND * 100
            NEXT I
        CASE IS < .7                             '0.3<=R<0.7ならば楕円模様
            FOR I = 0 TO 10
                CIRCLE (RND * 640, RND * 400), RND * 100,
                  RND * 10, -2 * 3.14 * RND, -2 * 3.14 * RND, 2 * RND
            NEXT I
        CASE IS < .9                               '0.7<=R<0.9ならば箱模様
            FOR I = 0 TO 10
                LINE (RND * 640, RND * 400)-
                                  (RND * 640, RND * 400), RND * 10, B
            NEXT I
        CASE ELSE                                  '0.9<=R<1ならば画面消去
            CLS
    END SELECT
LOOP
```

CASEは、CASE IS （式）で、式を書くことができます。

上のプログラムのように、CASE IS < .3 で選択されれば、次のCASE IS < .7 で選択されないので、0.3<=R<0.9 のような選択も可能になります。

また、例えばR=3,4,5,6,8,9 のように7が抜けた選択は、CASE 3 TO 6, 8 TO 9 と書きます。

2．35　手続きの独立

いよいよプログラムが大きくなって、次第にプログラが読みにくくなってきました。

そこで、わかりやすいように書き方を工夫しましょう。それには、プログラムをその働きのまとまりごとに独立させることです。独立させたプログラムのことをサブプロシージャ（サブの手続き）といいます。それに対して、親の手続きをメインプロシージャということができます。

```
'QB2_35_1 サブプロシージャの利用例
DECLARE SUB ENMOYOU ()              '円模様のサブプロシージャを使うという宣言
DECLARE SUB HAKOMOYOU ()            '箱模様のサブプロシージャを使うという宣言
DECLARE SUB MATIGAI () '間違いを知らせるサブプロシージャを使うという宣言
DO
    CLS
    PRINT "模様を選択して番号を入力してください。"
    INPUT "円模様:1 箱模様:2 終了:E "; BANGOU$
                                    'BANGOU$に1か2かEを受け入れる
    CLS
    SELECT CASE BANGOU$             'BANGOU$の内容によって仕事をする
        CASE "1"                    'BANGOU$が"1"の場合の仕事＝円模様
            ENMOYOU
             'サブプロシージャの名前ENMOYOUとすればENMOUYOUが動くのです
        CASE "2"                    'BANGOU$が"2"の場合の仕事＝箱模様
            HAKOMOYOU
        CASE "E"                    'BANGOU$が"E"の場合の仕事＝終了
            EXIT DO
        CASE ELSE                   'BANGOU$が1,2,E以外の場合の仕事
            MATIGAI
    END SELECT                      '選択の仕掛の終了
LOOP

SUB ENMOYOU                         '円模様のサブプロシージャ
    FOR i = 1 TO 400 STEP 5
        CIRCLE (i, i), i / 2, i MOD 7 + 1
    NEXT i
END SUB

SUB HAKOMOYOU                       '箱模様のサブプロシージャ
    FOR i = 1 TO 400 STEP 2
        LINE (i, i)-(i * 2, i * 2), i MOD 7 + 1, B
    NEXT i
END SUB
```

```
SUB MATIGAI                           '番号間違いを知らせるサブプロシージャ
    PRINT "番号違いです。"                '間違った入力をお知らせする
    BEEP: BEEP: BEEP                     '警告音を3回鳴らす
END SUB
```

最初のDECLAREステートメントは、入力する必要がありません。自動的に生成するようになっています。

SUB ENMOYOU[↵]と入力すると、突然画面が消えて、SUB ENMOYOU.....END SUBと表示されます。この間にプログラムを入力します。メインや他のサブプロシージャを表示するには、[SHIFT]を押しながら[f・2]を押します。

プロシージャ（手続き）は互いに独立しており、同じ名前の変数があっても影響しあうことはありません。同じ名前の変数を使って、容易にプログラミングができるようになっています。

サブプロシージャを上手に使うことが、美しくわかりやすいプログラムを作るための第1歩なのです。

2．３６　手続き間の手紙

手続きの名前を書くだけで、独立した手続きの機能を容易に利用できるのですが、これだけでは、独立した手続きには固定されたデータに基づいた動作以外させられなくなってしまいます。そこで、独立した手続きに対して注文の手紙を出して、都合よく動作するよう依頼することができます。

次の例は、メインプロシージャが、模様を変化させるデータをサブプロシージャENMOYOUへ送り、ENMOYOUはそのデータに基づいた円模様を描いてからサブプロシージャHAKOMOYOUへデータを送り、HAKOMOYOUはそのデータに基づいた箱模様を描くようになっています。

```
'QB2_36_1 プロシージャ間のデータの手紙
DECLARE SUB ENMOYOU (KIZAMI!, BAIRITU!)
DECLARE SUB HAKOMOYOU (HAZIME!, KIZAMI!, BAIRITU!)
CLS
FOR I = 5 TO 100 STEP 5            '円の半径を変化させるきざみを生成する
    FOR J = .1 TO .9 STEP .2
                                   '円の半径を変化させる倍率を生成する
        CLS                        '画面消去
        ENMOYOU I, J               'ENMOYOUへきざみと倍率を送る
    NEXT J
NEXT I
END

SUB ENMOYOU (KIZAMI, BAIRITU)                   'サブプロシージャ　ENMOYOU
                                                'きざみと倍率を受け取る
FOR I = 1 TO 600 STEP KIZAMI                    '円の中心の座標を1から600ま
                                                'で等間隔のきざみで生成する
    CIRCLE (I, I), I * BAIRITU, I MOD 7 + 1              '円を描く
    HAKOMOYOU I, KIZAMI / 2, BAIRITU / 3        'HAKOMOYOUへ開始位置、
                                                'きざみ、倍率を送る
NEXT I
END SUB

SUB HAKOMOYOU (HAZIME, KIZAMI, BAIRITU)         'サブプロシージャ HAKOMOYOU
                                       '開始位置、きざみ、倍率を受け取る
FOR I = HAZIME TO HAZIME + 100 STEP KIZAMI
                                                '箱模様を描く位置を生成する
    LINE (I, I)-((I + HAZIME) * BAIRITU,
             (I + HAZIME) * BAIRITU), I MOD 7 + 1, B     '箱を描く
NEXT I
END SUB
```

ENMOYOU I,J　のI,Jは、ENMOYOUへ送るデータです。（これを実引数といいます。）

ENMOYOUはこのデータを、SUB ENMOYOU (KIZAMI, BAIRITU)の(KIZAMI, BAIRITU)で受け取ります。（これを仮引数といいます。）すると、IのデータがKIZAMIへ、JのデータがBAIRITUへ送られます。ENMOYOUは、この送られたデータを使って動作します。

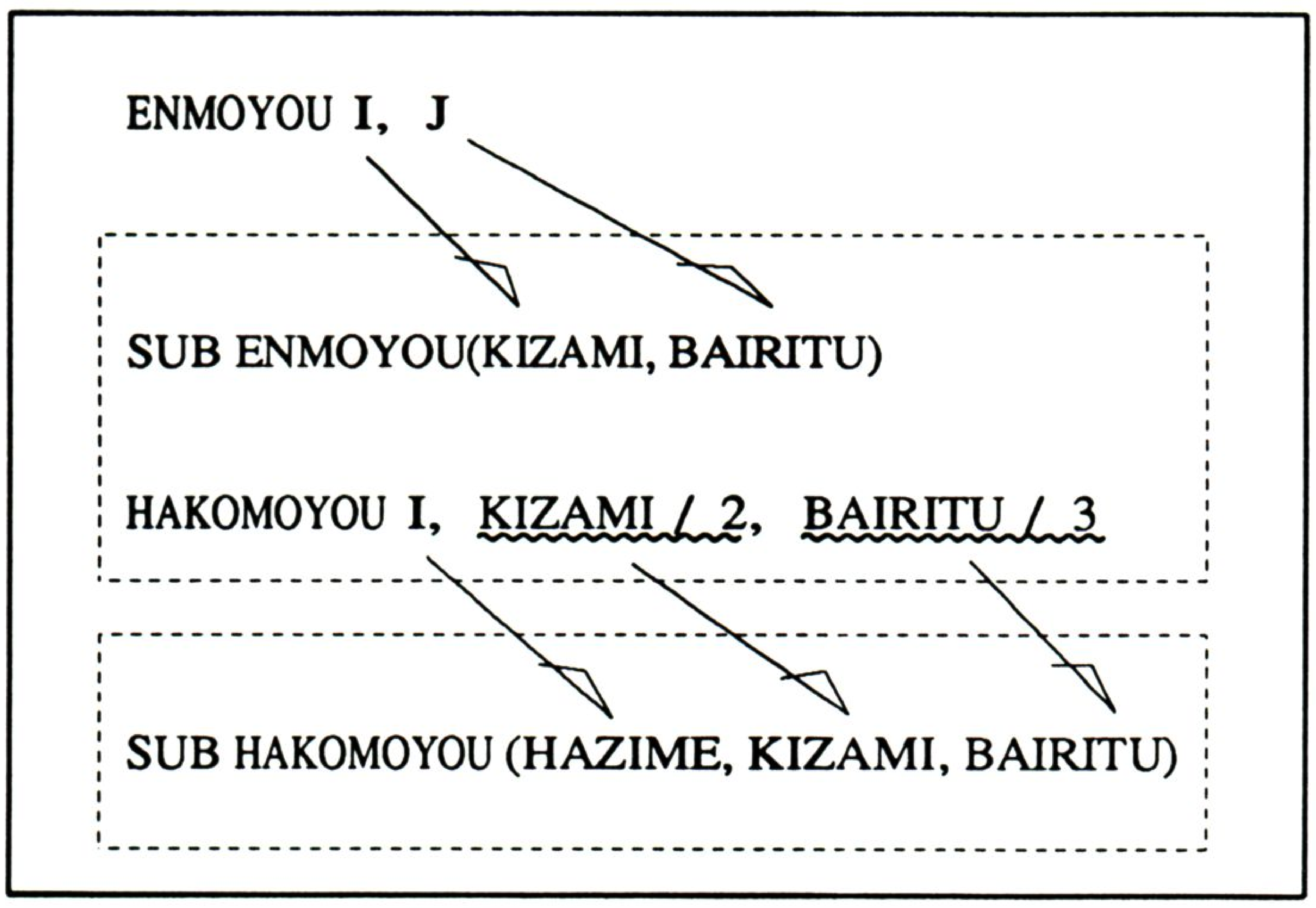
ENMOYOUは、HAKOMOYOU I, KIZAMI / 2, BAIRITU / 3 として、データをHAKOMOYOUに送ります。

HAKOMOYOUは、ENMOYOUのIをHAZIMEに、KIZAMI / 2 をKIZAMIへ、BAIRITU / 3をBAIRITUへ受け取り、このデータを使って動作します。

送り側の実引数には、変数名、式、数値のいずれも使用できます。

変数名は、ここではわかりやすいようにメインとサブとで同じ変数名を使用していますが、別の変数として取り扱っており、互いに独立しています。

また、この例にはないのですが、サブプロシージャの計算結果をメインへ送ることもできます。



第❸章

アニメーション

アニメーションの技法を楽しみながら学び
ゲームつくりに挑戦します。

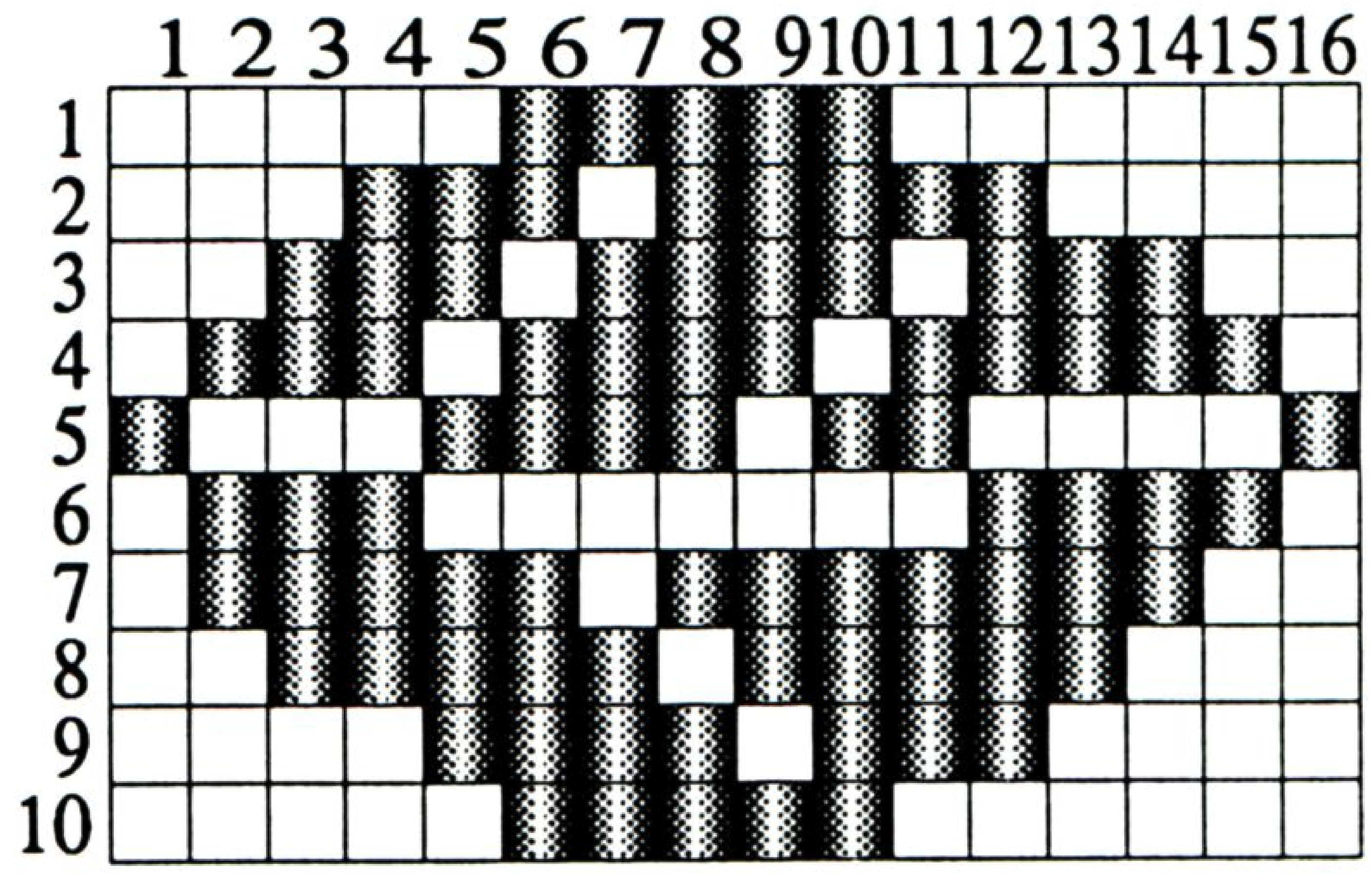
3．1　キャラクタは点点

アニメーションを作るには、まず、動かすためのキャラクタを用意します。

あの有名なマリオのオジサンをはじめキャラクタはすべて点からできていますから、キャラクタ作りには根気がいります。さらに、キャラクタのデザインがよくなければなりませんから、センスを必要とします。

木の葉のキャラクタを設計してみました。これを画面表示するには、PSETステートメントを使います。

```
'QB3_1_1　アニメ1
CLS
PSET (5, 0), 2
PSET (6, 0), 2
PSET (7, 0), 2
PSET (8, 0), 2
以下、点の数だけPSETを繰り返す
```



しかし、点の数だけPSETを繰り返していたのでは、あまりにもプログラムが大きくなりすぎます。そこで、少し工夫しました。

```
'QB3_1_2 アニメ2
CLS
FOR Y = 1 TO 10                              '縦方向の座標を決める
    FOR X = 1 TO 16                          '横方向の座標を決める
        READ C                  'データ（色番号）を読んで、Cに入れる
        PSET (X, Y), C                       '点を描く
    NEXT X
NEXT Y
END
DATA  0,0,0,0,0,2,2,2,2,2,0,0,0,0,0,0        '木の葉の色番号のデータ
DATA  0,0,0,2,2,2,0,2,2,2,2,2,0,0,0,0
DATA  0,0,2,2,2,0,2,2,2,2,0,2,2,2,0,0
DATA  0,2,2,2,0,2,2,2,2,0,2,2,2,2,2,0
DATA  2,0,0,0,2,2,2,2,0,2,2,0,0,0,0,2
DATA  0,2,2,2,0,0,0,0,0,0,0,2,2,2,2,0
DATA  0,0,2,2,2,2,0,2,2,2,2,2,2,2,0,0
DATA  0,0,2,2,2,2,2,0,2,2,2,2,2,0,0,0
DATA  0,0,0,2,2,2,2,2,0,2,2,2,0,0,0,0
DATA  0,0,0,0,0,2,2,2,2,2,0,0,0,0,0,0
```

READ Cは、プログラムの中に書いてあるデータを読むステートメントです。データには、DATAとかいてあります。

0,2,0,0,2,2 の中のカンマは、データの区切り記号です。READ Cで区切りと区切りの間の一つのデータを読み込んで、Cという変数に入れるのです。数字は色番号で、0は黒色、2は緑色ということです。

３．２　キャラクタをつくるコツ

QB3_1_2のプログラムでもよいのですが、木の葉の色番号のデータが少し見にくいので、次のように改善して、キャラクタを入力しやすく分りやすいよう改善しました。

```
'QB3_2_1 アニメ３
CLS
FOR Y = 1 TO 10                          '縦方向の座標を決める
    READ A$
          'キャラクタのデータを１行分読み取って変数A$に入れる
    FOR X = 1 TO 16                      '横方向の座標を決める
        PSET (X, Y), VAL (MID$ (A$, X, 1))          '点を描く
    NEXT X
NEXT Y
END
DATA .....22222......                    '木の葉の色番号のデータ
DATA ...222.22222....
DATA ..222.2222.222..
DATA .222.2222.22222.
DATA 2...2222.22....2
DATA .222.......2222.
DATA ..2222.2222222..
DATA ..22222.22222...
DATA ...22222.222....
DATA .....22222......
```

READ A$は、A$という変数に、DATAに続く１行分の　.....22222......　をすべて読み込むのです。

A$のように$のマークがついている変数は、文字型変数といって、アルファベットも数字も漢字もすべて文字として読み込んでしまいます。

A$="Quick BASIC"、A$="楽しいクイックベーシック"、A$="...22222.222...."などと、文字列を記憶でき、画面表示は、PRINT A$でよいのです。

A$に記憶された文字の一部を取り出して使うこともできます。取り出すためには、MID$という関数を道具として利用します。たとえば、A$には"Quick BASIC"という文字列が記憶されていて、B$=MID$(A$,5,1)とすれば、A$の5文字目から1文字分の"k"が、B$に記憶されます。

例を示します。

A$="Quick BASIC"	B$=MID$(A$,7,3)		
	A$の7文字目から3文字	B$の中は"BAS"	
A$="COMPUTER"	B$=MID$(A$,1,4)		
	A$の1文字目から4文字	B$の中は"COMP"	
A$="...22222.222...."	B$=MID$(A$,X,1)		
	A$のX文字目から1文字		
		X=5ならば、B$の中は"2"	

なお、文字列が漢字の場合は、KMID$という関数を使います。

このようにして、A$の文字列の中から１文字ずつ取り出すことはできましたが、"2"を取り出してもこれは文字の2なので、色番号として利用するためにはさらにこれを数値の2に変換する必要があります。

文字を数値に変換するには、VALという関数を使います。たとえば、C=VAL("2")とすれば、Cには数値の2が記憶されます。

例を示します。

A$="123"	C=VAL(A$)	Cの中は、数値の123
A$="6428"	C=VAL(A$)	Cの中は、数値の6428
A$="2"	C=VAL(A$)	Cの中は、数値の2

ところが、葉のキャラクタのデータには、ピリオド"."があります。これは数値ではないので、VAL関数は、読み込みを中止します。そこで、結局、黒の色番号の　0（ゼロ）が計算されるわけです。

このようにして、PSET (X, Y), VAL (MID$ (A$, X, 1)) のうち、
VAL (MID$ (A$, X, 1))が、色番号を計算しているのです。

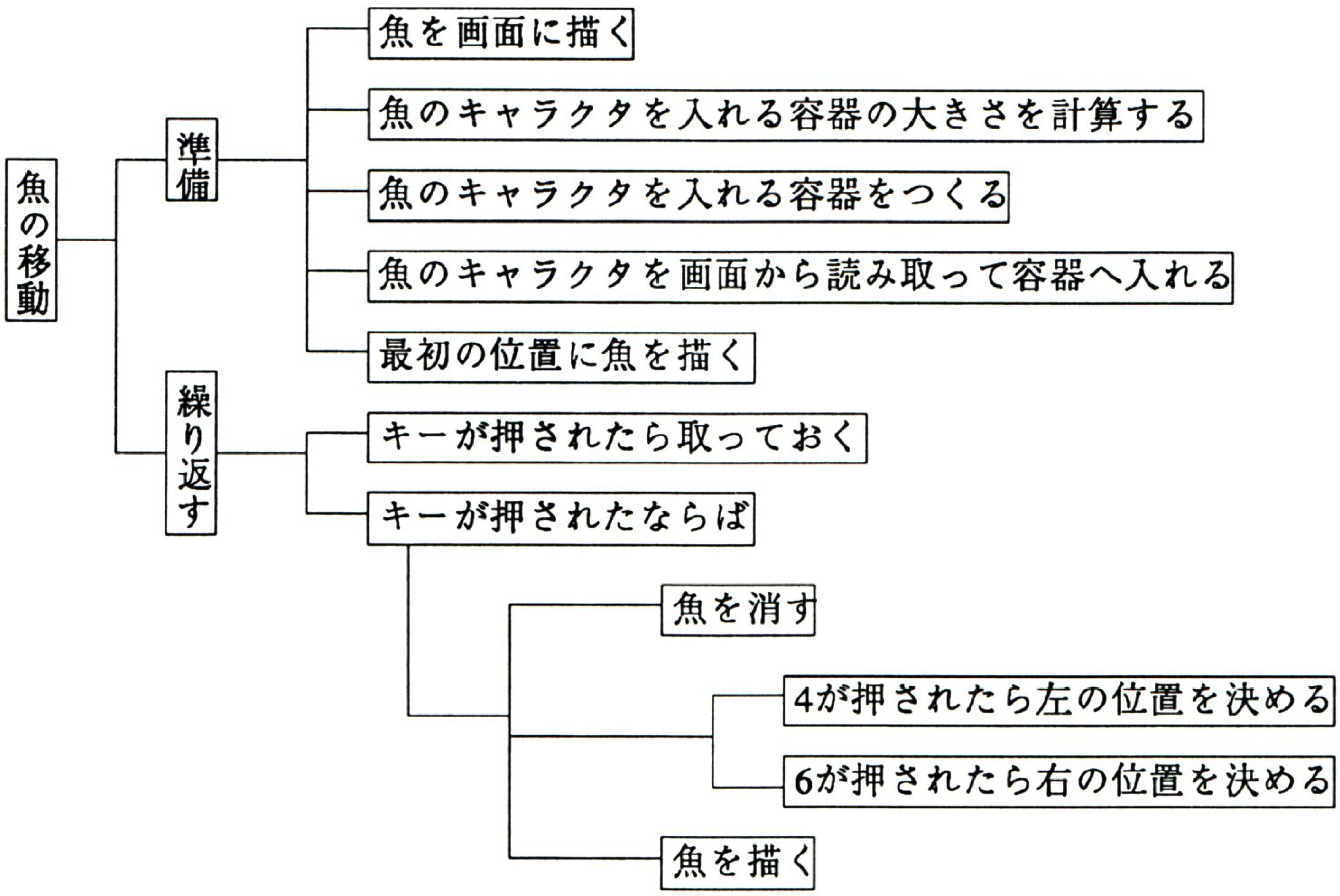
PSET (X,　Y)...... の(X,Y)は、画面上の座標です。これは、キャラクタのデータの位置に丁度対応しているので、便利なのです。

3．3　魚は泳ぐ

さて、いよいよ、魚のキャラクタをつくって、泳がせましょう。

このプログラムは、魚のキャラクタをつくり、魚を画面中央の(300,200)の位置に描き、テンキーボードの4を押すと、魚は左へ、6を押すと右へ、50ピクセル分移動します。

そのために必要とする仕事は次の図のとおりです。




```
'QB3_3_1 アニメ4　魚の移動
DEFINT T,X-Y                            '整数型変数名を決める
CLS                                     '画面消去
TATE = 17: YOKO = 48                    'キャラクタの縦横の大きさを決める
FOR Y = 1 TO TATE                       'キャラクタのデータを読んで
                                                      '画面に表示する
    READ A$
    FOR X = 1 TO YOKO
        PSET (X, Y), VAL(MID$(A$, X, 1))
    NEXT X
NEXT Y
OOKISA = (4 + 4 * ((YOKO + 7) ¥ 8) * TATE) ¥ 2
                 '画面から読み取ったキャラクタを入れる容器の大きさを計算する
DIM SAKANA(1 TO OOKISA) AS INTEGER      '画面から読み取ったキャラクタを
                                                   '入れる容器をつくる
GET (1, 1)-(YOKO, TATE), SAKANA         'キャラクタを画面から読み取る
```

```
X = 300: Y = 200                             '最初に魚を表示する位置を決める
PUT (X, Y), SAKANA, XOR                      '最初の位置へ魚を表示する
DO                                           '繰り返す
    K$ = INKEY$                             'キーが押されれば1文字をK$へ入れる
    IF K$ <> "" THEN                         'キーが押されていれば
        PUT (X, Y), SAKANA, XOR              '魚を消し
        SELECT CASE K$
            CASE "4"                         '4が押されていれば
                X = X - 50                                  '50ピクセル分左へ
            CASE "6"                         '6が押されていれば
                X = X + 50                                  '50ピクセル分右へ
        END SELECT
        PUT (X, Y), SAKANA, XOR              '新しい位置へ魚を描く
    END IF
LOOP
END

DATA ................22222222.......................
DATA ..............222222222222....................3
DATA ...........333333333333333333................33
DATA ........33333333333333333333333333.........3333
DATA ......33333333333333333333333333333333.....33333
DATA ...333333.3333333333333333333333333333333333333.
DATA .333333333.333333333333333333333333333333333333.
DATA 33333333333.333333333333333333333333333333333..
DATA 333....33333.33322222222222222222222222222233333..
DATA 333....33333.33335555555555555555555555553333333.
DATA 33333333333.333333335555555555555333333333333333.
DATA .33333333333333333333333333333..........333333
DATA ...3333333.33...33333333333333..............3333
DATA ......3333.333333..33333333..................33
DATA ...........33333333.........................3
DATA ..............3333...........................
DATA ..............33.............................
```

アニメは、画面から読み取ったキャラクタの表示と消去を繰り返すことによって動きます。

まず、画面から読み取ったキャラクタを入れておく変数の大きさを計算します。この計算式は次のとおりです。

OOKISA = (4 + 4 * ((YOKO + 7) ¥ 8) * TATE) ¥ 2

キャラクタの縦と横の幅（ピクセルの数）をこの計算式のTATEとYOKOにセットすれば計算できます。

次に、キャラクタを入れる容器をつくります。

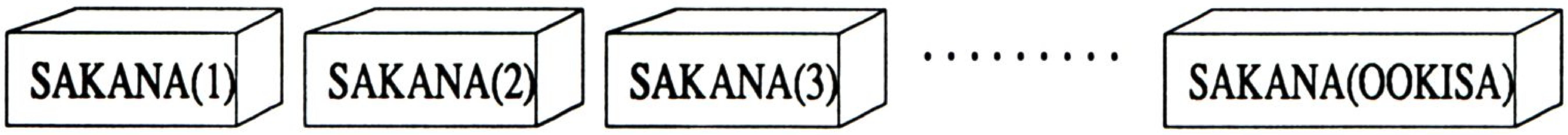
DIM SAKANA(1 TO OOKISA) AS INTEGER

一つの変数だけでは入らないくらい大量のデータなので、SAKANAという整数の変数（AS INETGER）をOOKISA分の数だけ用意します。

DIM（DIMESIONの略）は、大きさとか次元という意味です。

そこで、DIM SAKANA(1 TO OOKISA) AS INTEGER は、「大きさは、SAKANAの名前で整数型の変数を1番からOOKISA番までの番号を付けて用意します」という意味です。一般にこのような変数の設定の仕方を、「配列」といいます。

配列は、例えば、SAKANA(1),SAKANA(2)...のように、番号が付いた変数として、取り扱うことができます。



次に、画面からキャラクタをこの容器へ読み取って入れます。

GET (1, 1)-(YOKO, TATE), SAKANA

GETは得るという意味ですから、画面の(1, 1)-(YOKO, TATE)の範囲の画面を、SAKANAという変数に取り入れるということです。SAKANAは、番号が付いた変数ですが、ここでは番号を付けないで使うようになっています。

これで、準備は終了しました。いよいよ、アニメを描く仕事です。

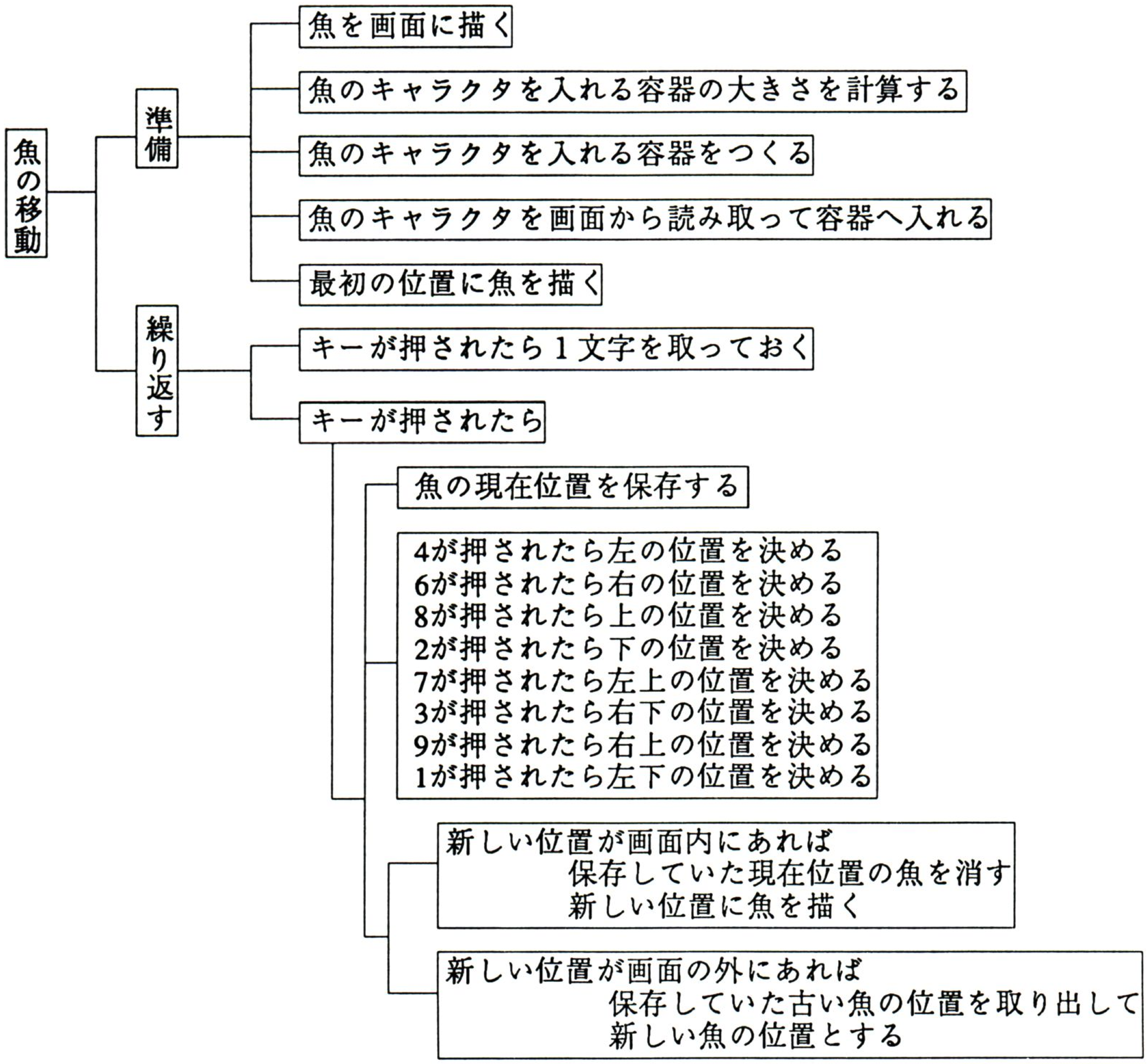
SAKANAの入っているキャラクタは、PUT (X, Y), SAKANA, XOR で描きます。PUTは置くという意味ですから、画面上の(X,Y)の位置へSAKANAのキャラクタを置く（描く）ということです。

さて、XORは何でしょうか。これは画面の表示の状態と今から描こうとしているものとを較べて、排他的論理和で調整して描くということです。XORの外に、PSETなどあるのですが、XORを利用する理由は、もう一度同じ位置に同じ方法で描くとキャラクタが消えるからです。これはアニメを描くには都合のよい機能です。そこで、キャラクタを消す場合も同じステートメントを使えるのです。

3．4　画面内を泳ぐ

前節のアニメ４のプログラムを実行すると、すぐに「引数が許される範囲ではありません（ERR=5）」とエラー表示がでます。それは、画面の両端にキャラクタを移動した場合のことを考えていなかったからです。

そこで今度は、テンキーを利用して上下左右と斜めに移動できるようにし、同時に、キャラクタが画面の上下左右をはみ出ないように改善します。このプログラムの構造は次のとおりです。



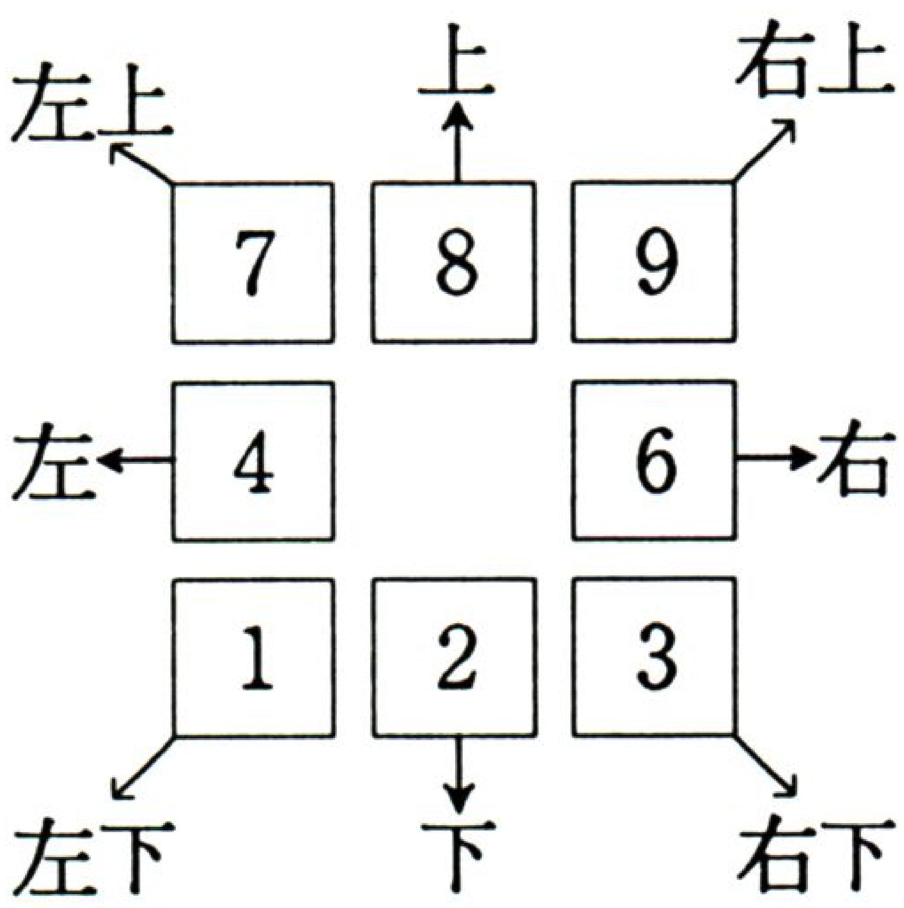
```
'QB3_4_1 アニメ５　魚が画面の範囲内を移動する
DEFINT T, X-Y                          '整数型変数名を決める
CLS                                    '画面消去
TATE = 17: YOKO = 48                   'キャラクタの縦横の大きさを決める
FOR Y = 1 TO TATE                'キャラクタのデータを読んで画面に表示する
    READ A$
    FOR X = 1 TO YOKO
        PSET (X, Y), VAL(MID$(A$, X, 1))
    NEXT X
NEXT Y
OOKISA = (4 + 4 * ((YOKO + 7) ¥ 8) * TATE) ¥ 2
            '画面から読み取ったキャラクタを入れる容器の大きさを計算する
DIM SAKANA(1 TO OOKISA) AS INTEGER
                     '画面から読み取ったキャラクタを入れる容器を作る
GET (1, 1)-(YOKO, TATE), SAKANA        'キャラクタを画面から読み取る
CLS                                    '画面消去
X = 300: Y = 200                       '最初に魚を表示する位置を決める
PUT (X, Y), SAKANA, XOR                '最初の位置へ魚を表示する
DO                                     '繰り返す
    K$ = INKEY$                        'キーが押されれば１文字をK$へ入れる
    IF K$ <> "" THEN                   'キーが押されていれば
        HX = X: HY = Y
                                       '魚の現在位置をHX, HYに保管しておく
        S = 50                         '１回に移動するピクセル数のセット
        SELECT CASE K$
            CASE "4"                               '4が押されていれば
                X = X - S                              '左へ
            CASE "6"                               '6が押されていれば
                X = X + S                              '右へ
            CASE "8"                               '8が押されていれば
                Y = Y - S                              '上へ
            CASE "2"                               '2が押されていれば
                Y = Y + S                              '下へ
            CASE "7"                               '7が押されていれば
                X = X - S: Y = Y - S                   '左上へ
            CASE "3"                               '3が押されていれば
                X = X + S: Y = Y + S                   '右下へ
            CASE "9"                               '9が押されていれば
                X = X + S: Y = Y - S                   '左上へ
            CASE "1"                               '1が押されていれば
                X = X - S: Y = Y + S                   '右下へ
        END SELECT
        IF (X >= 0) AND (Y >= 0) AND (X <= 639 - YOKO) AND
          (Y <= 399 - TATE) THEN       '魚の位置が画面の範囲内ならば
            PUT (HX, HY), SAKANA, XOR              '魚を消す
            PUT (X, Y), SAKANA, XOR                '新しい位置へ魚を描く
        ELSE                           '魚の位置が画面の範囲外なので
            X = HX: Y = HY                         '範囲内に戻す
```

```
            END IF
        END IF
LOOP
END
```

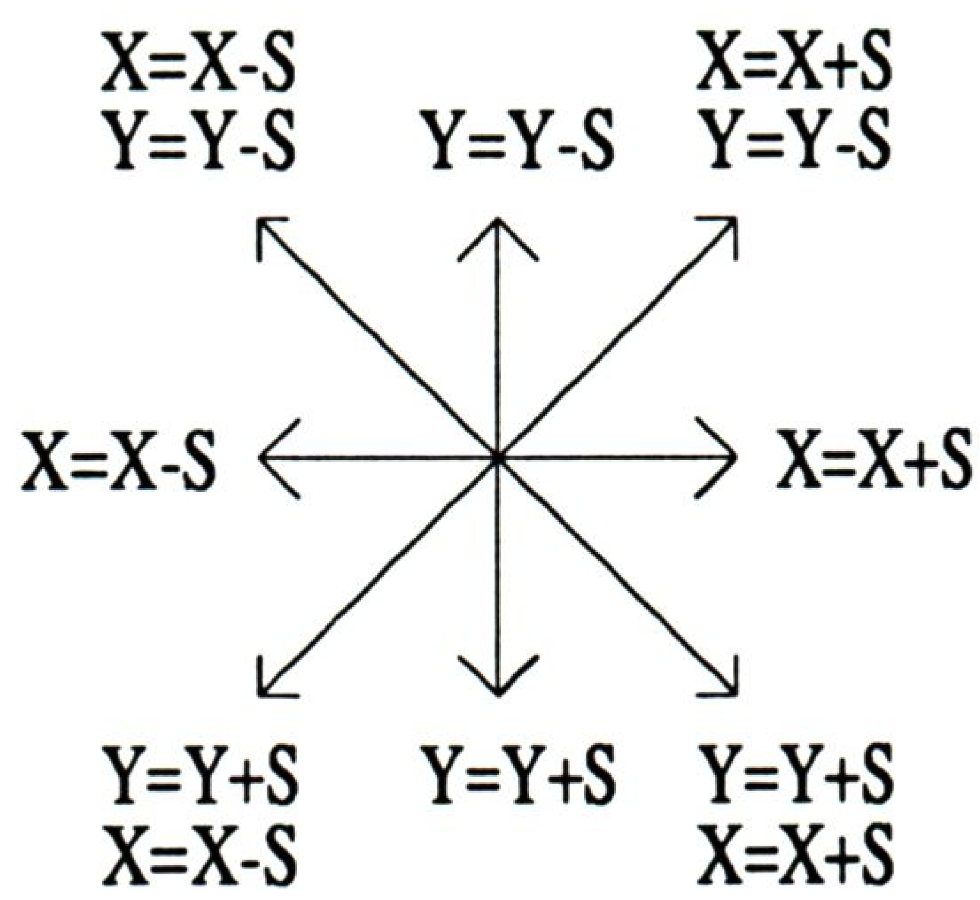
（以下の魚のキャラクタのデータは省略）

魚の左右の移動はXに移動数（S=50）を加減し、上下の移動はYに移動数（S=50）を加減して行い、また、斜めの移動はXとYに同時に移動数（S=50）を加減して行っています。

テンキーの押し方



画面上の座標の動き



画面の範囲の内外の判定は、キャラクタの位置XとYの値で行っていますが、キャラクタには厚みがあるので、画面の右と下についてはTATEとYOKOを減じています。

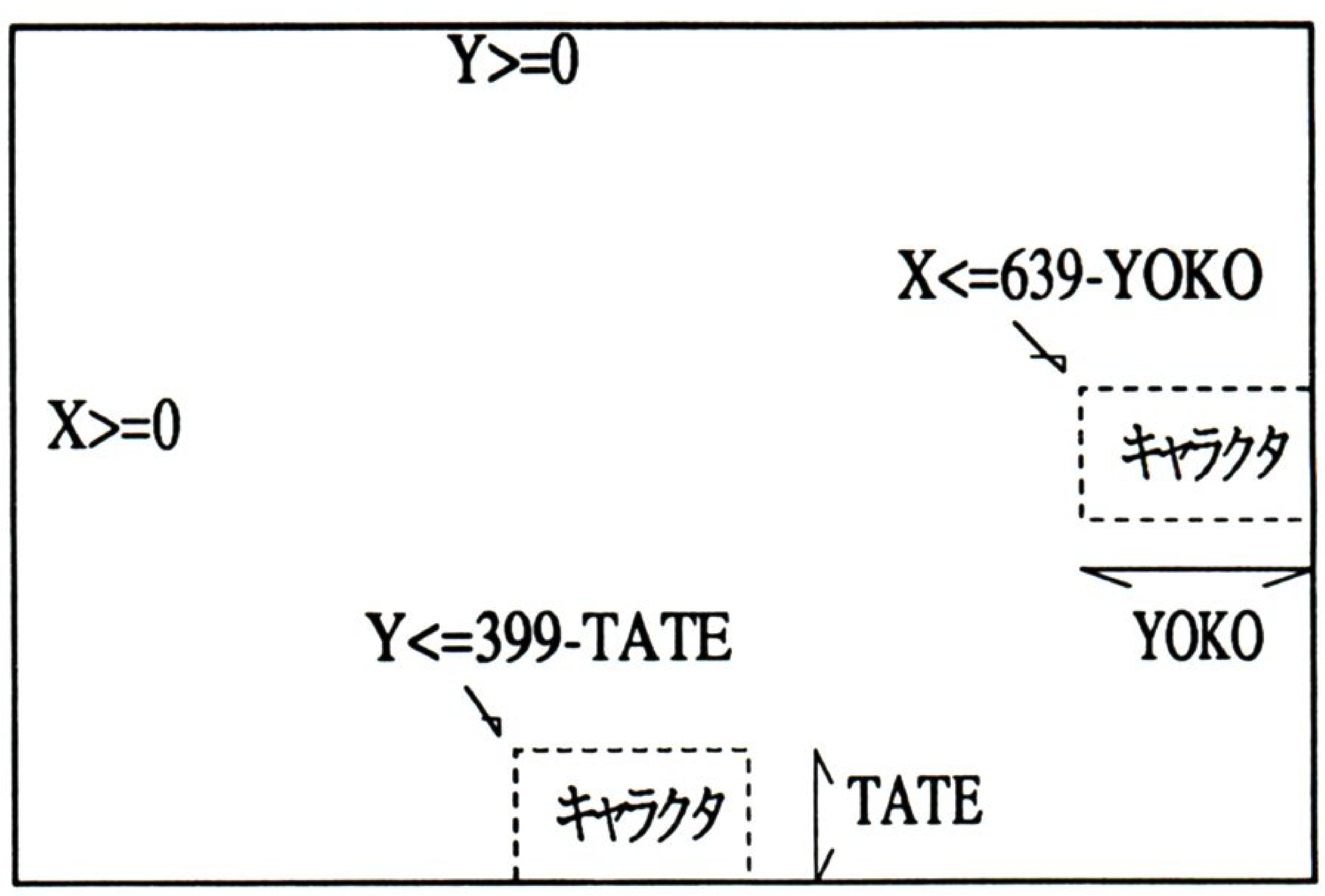
IF (X >= 0) AND (Y >= 0) AND (X <= 639 - YOKO) AND (Y <= 399 - TATE) THEN

ANDは、XとYのそれぞれの条件が同時に成立することを条件としていることです。

ここでは「(X >= 0)及び(Y >= 0)及び(X <= 639 - YOKO)及び(Y <= 399 - TATE)が同時に成立する（魚が画面の範囲内にある）ならば・・（通常の動作をする）」ということになります。

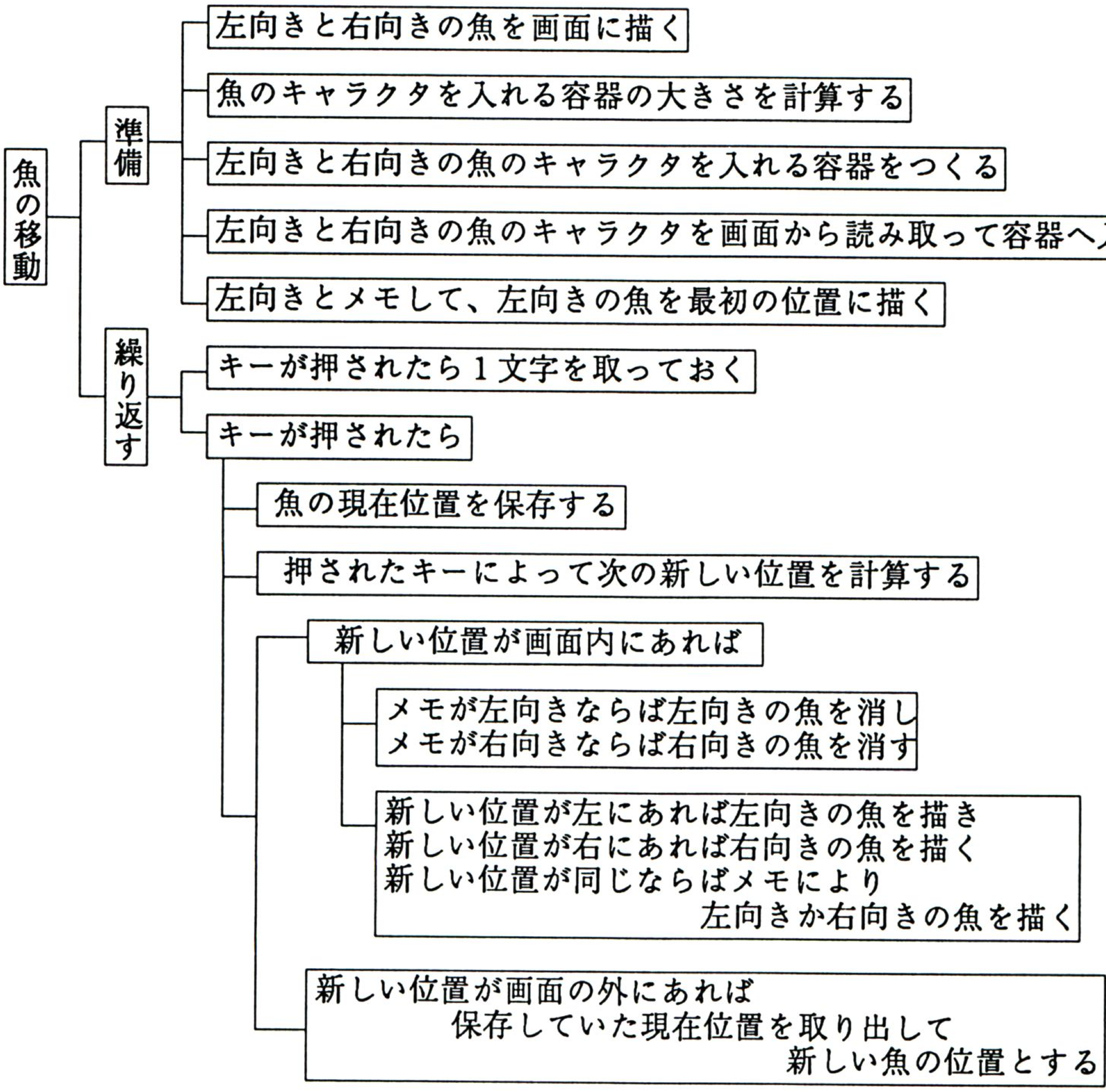
すぐ次の ELSE は、条件に合わない（魚が画面の範囲外にある）場合の動作（以前に保存していたキャラクタの位置のデータを元に戻す）を指示しています。

画面の範囲内外の判定



3．5　向きを変える魚

すでに気が付いていると思いますが、魚が後向きに移動しますね。そこで、今度は右向きの魚のキャラクタも用意して、必要に応じて左向きの魚と右向きの魚を使い分けるようにしましょう。プログラムの構造は次のとおりです。




```
'QB3_5_1 アニメ６　魚の向きを変える
DEFINT　H,M,O,S-U,X-Y                              '整数型変数名を決める
CLS                                                '画面消去
TATE = 17: YOKO = 48                         'キャラクタの縦横の大きさを決める
FOR Y = 1 TO TATE                     'キャラクタのデータを読んで画面に表示する
    READ A$
    FOR X = 1 TO YOKO
        PSET (X, Y), VAL(MID$(A$, X, 1))                     '左向きの魚を描く
        PSET (YOKO - X + 1, Y + TATE), VAL(MID$(A$, X, 1))
                                                             '右向きの魚を描く
    NEXT X
```

```
NEXT Y
OOKISA = (4 + 4 * ((YOKO + 7) ¥ 8) * TATE) ¥ 2
               '画面から読み取ったキャラクタを入れる容器の大きさを計算する
DIM SAKANA(1 TO OOKISA) AS INTEGER '左向きのキャラクタを入れる容器を作る
DIM SAKANA2(1 TO OOKISA) AS INTEGER '右向きのキャラクタを入れる容器を作る
GET (1, 1)-(YOKO, TATE), SAKANA    '左向きのキャラクタを画面から読み取る
GET (1, 1 + TATE)-(YOKO, TATE * 2), SAKANA2
                                   '右向きのキャラクタを画面から読み取る
CLS                                  '画面消去
X = 300: Y = 200                     '最初に魚を表示する位置を決める
MUKI$ = "HIDARI"                     '最初の向きをメモしておく
HIDARI = 0                           '移動範囲の画面の左端
MIGI = 639 - YOKO                    '移動範囲の画面の右端
UE = 0                               '移動範囲の画面の上端
SITA = 399 - TATE                    '移動範囲の画面の下端
PUT (X, Y), SAKANA, XOR              '最初の位置へ魚を表示する
DO                                   '繰り返す
    K$ = INKEY$                    'キーが押されれば1文字をK$へ入れる
    IF K$ <> "" THEN               'キーが押されていれば
        HX = X: HY = Y               '魚の現在位置をHX, HYに保管する
        S = 50                       '1回に移動するピクセル数をセット
        SELECT CASE K$
            CASE "4"                 '4が押されていれば
                X = X - S                     '左へ
    (前節のアニメ5のプログラムに同じなので一部省略します。)
            CASE "1"                 '1が押されていれば
                X = X - S: Y = Y + S          '右下へ
        END SELECT
        IF (X >= HIDARI) AND (Y >= UE) AND (X <= MIGI) AND
          (Y <= SITA) THEN           '魚の位置が画面の範囲内ならば
            SELECT CASE MUKI$
                CASE "HIDARI"                 '左向きならば
                    PUT (HX, HY), SAKANA, XOR '左向きの魚を消す
                CASE "MIGI"                   '右向きならば
                    PUT (HX, HY), SAKANA2, XOR '右向きの魚を消す
            END SELECT
            SELECT CASE X
                CASE IS < HX                  '左へ移動ならば
                    MUKI$ = "HIDARI"          'HIDARIとメモし
                    PUT (X, Y), SAKANA, XOR   '新しい位置へ
                                              '左向きの魚を描く
                CASE IS > HX                  '右へ移動ならば
                    MUKI$ = "MIGI"            'MIGIとメモし
                    PUT (X, Y), SAKANA2, XOR  '新しい位置へ
                                              '右向きの魚を描く
                CASE IS = HX                  '移動しないならば
                    SLECT CASE MUKI$
                        CASE "HIDARI"         '現在が左向きならば
                            PUT (X, Y), SAKANA, XOR '左向きの魚を描き
```

```
                    CASE "MIGI"                          '現在が右向きならば
                        PUT (X, Y), SAKANA2, XOR         '右向きの魚を描く
                END SELECT
            END SELECT
        ELSE                                    '魚の位置が画面の範囲外なので
            X = HX: Y = HY                               '範囲内に戻す

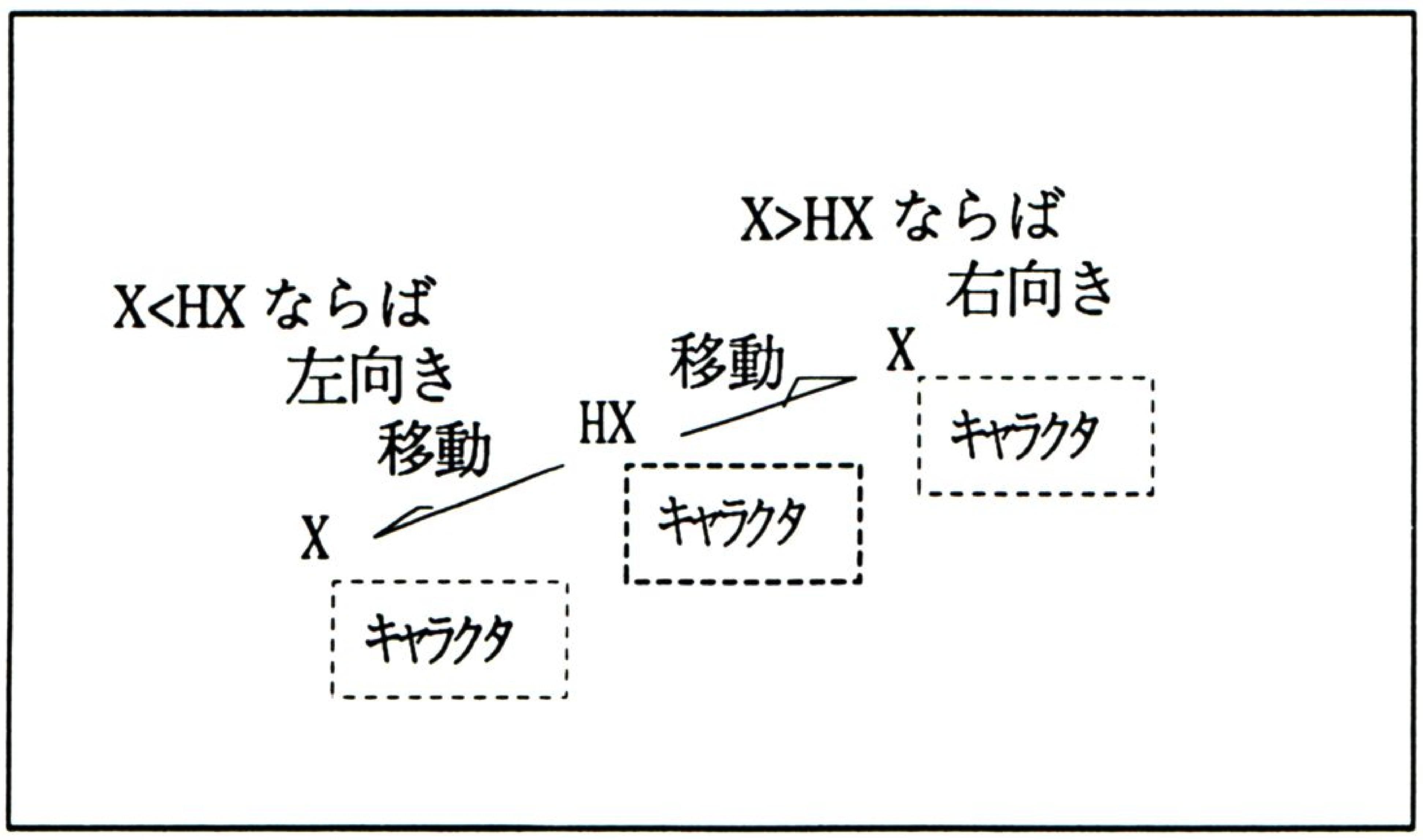
        END IF
    END IF
LOOP
END
```
（以下の魚のキャラクタのデータを省略します。）

左向きと右向きの魚のキャラクタを同時に表示し、左向きをSAKANAへ、右向きをSAKANA2へ読み込むようにしました。

現在どちらのキャラクタを使用しているか分るように、MUKI$というメモを用意しています。

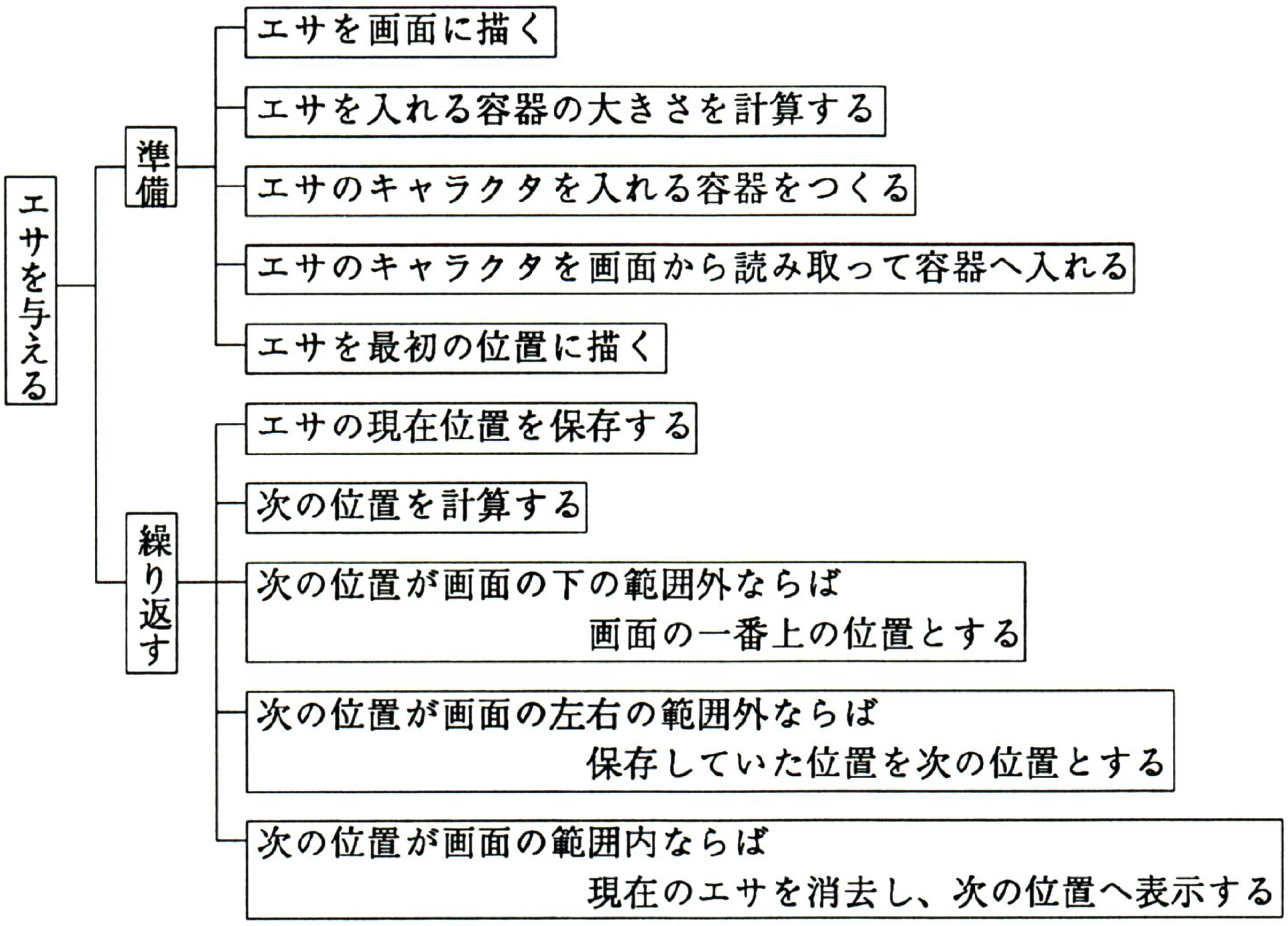
キャラクタを消去するときは、このメモの向きのキャラクタをPUTします。また、現在の位置HXと新しい位置Xとを比較して、移動する方向によってキャラクタを使い分けるようになっています。

左向き右向きの選択



3．6　エサを与える

魚ができましたので、今度は魚にエサを与えたいと思います。まずはじめに、1個のエサを与えるプログラムをつくります。プログラムの構造は次のとおりです。



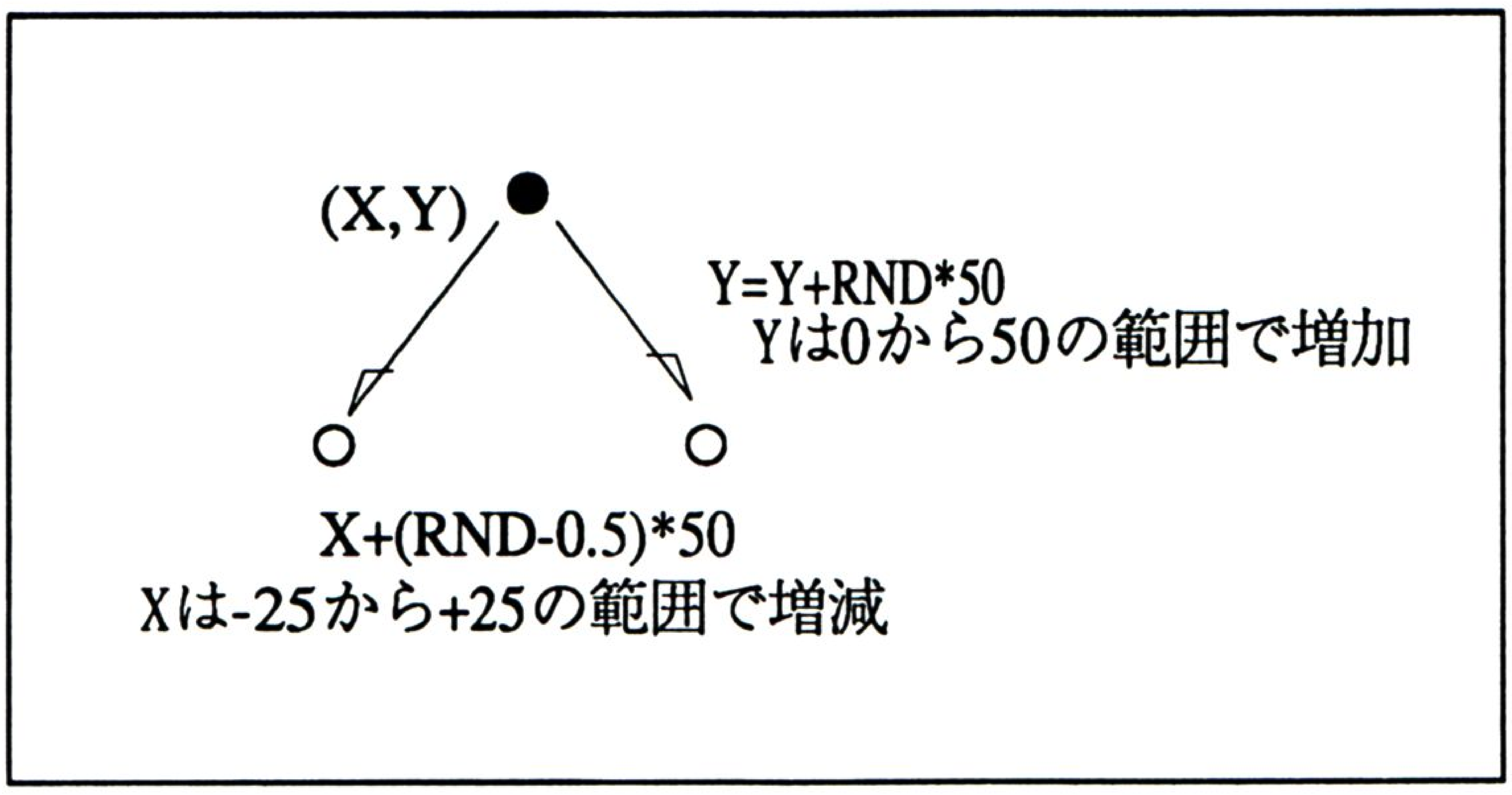

```
QB3_6_1 'アニメ７　エサを与える
DEFINT  X-Y                                  '整数型変数の設定
RANDOMIZE  TIMER                             '乱数系列の設定
CLS                                          '画面消去
FOR Y = 1  TO 5                              'エサのキャラクタを描く
    READ A$
    FOR X = 1  TO 5
        PSET  (X,  Y),  VAL(MID$(A$, X,  1))
    NEXT X
NEXT Y
OOKISA  = (4  + 4  *  ((5  + 7) ¥ 8)  * 5) ¥ 2
                          'エサのキャラクタを入れる容器の大きさを計算する
DIM  ESA(1 TO OOKISA)  AS  INTEGER     'エサのキャラクタを入れる容器を作る
GET  (1,   1)-(5,  5),  ESA                  'エサを読み取って容器へ入れる
CLS                                          '画面消去
X = 300:  Y = 0                              '最初の表示位置
PUT  (X,  Y), ESA, XOR                       '最初の表示
```

```
DO                                        '繰り返す
    HX = X: HY = Y                        '現在の位置を保存する
    X = X + (RND - .5) * 50               '次の位置を計算する
    Y = Y + RND * 50                      '次の位置を計算する
    IF Y > 394 THEN                       '次の位置が画面下の外ならば
        Y = 0                                 '画面の一番上へ
    END IF
    IF (X < 0) OR (X > 634) THEN          '画面の両端の外ならば
        Y = HY: X = HX                 '保存していた位置を次の位置に戻す
    ELSE                                  '画面の範囲内ならば
        PUT (HX, HY), ESA, XOR                '現在のエサを消去する
        PUT (X, Y), ESA, XOR                  '次のエサを描く
    END IF
LOOP
END

DATA .666.                                'エサのキャラクタのデータ
DATA 66666
DATA 66666
DATA 66666
DATA .666.
```

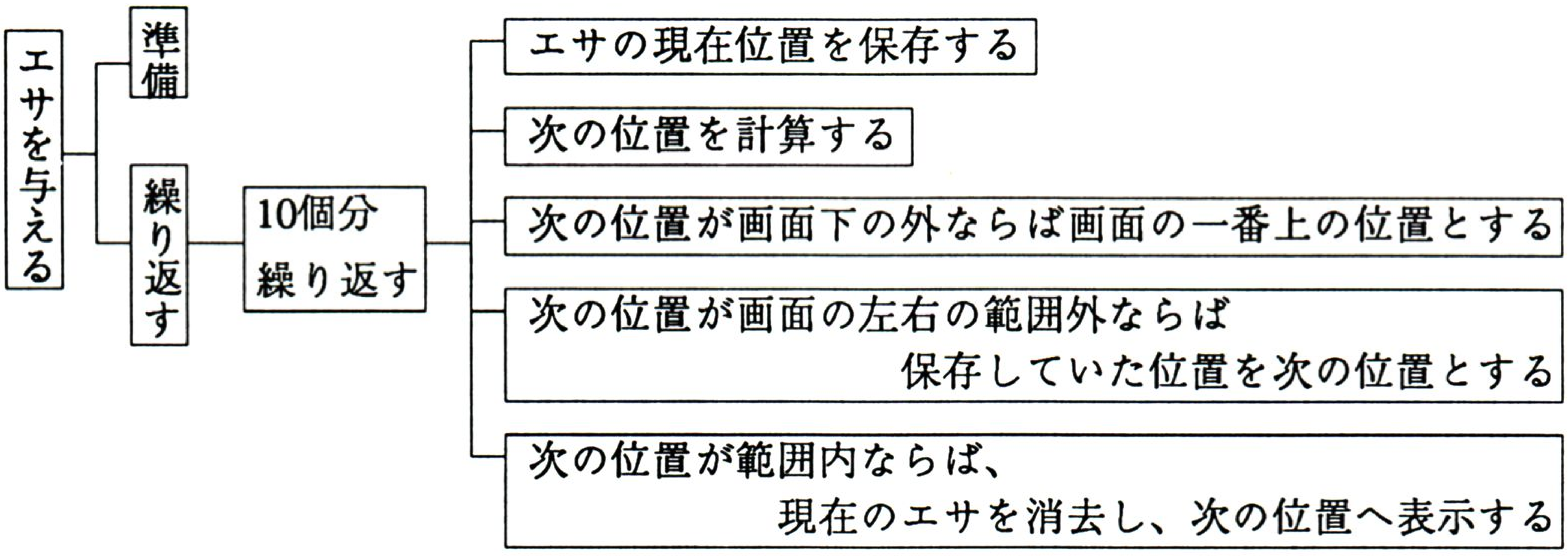
X = X + (RND - 0.5) * 50 は、エサの次の横位置の計算です。RNDは乱数で0と1の間の一様乱数なので、RND * 50は、-0.5から+0.5の間の数になります。これに50を乗ずるので、結局左右に25ピクセルの範囲で移動することになります。

また、Y = Y + RND * 50 の計算式は、下の方に向かって0から50ピクセルの範囲で移動する位置を計算します。



3．7　たくさんのエサ

たった１個のエサでは魚は生存できませんから、たくさんのエサを与えるよう改善します。前節のプログラムを１０個繰り返しては冗長すぎるので、工夫しました。プログラムの構造は次のとおりです。




```
DECLARE SUB ESAMAKI (ESA%(), X%, Y%, HX%, HY%)
'アニメ８ たくさんのエサを与える
DEFINT H-I, X-Y                                  '整数型変数の設定
RANDOMIZE TIMER                                  '乱数系列の設定
CLS                                              '画面消去
FOR Y = 1 TO 5                                   'エサのキャラクタを画面に描く
    READ A$
    FOR X = 1 TO 5
        PSET (X, Y), VAL(MID$(A$, X, 1))
    NEXT X
NEXT Y
OOKISA = (4 + 4 * ((5 + 7) ¥ 8) * 5) ¥ 2
                                  'エサのキャラクタの容器の大きさを計算する
DIM ESA(1 TO OOKISA) AS INTEGER                  'エサのキャラクタの容器を作る
GET (1, 1)-(5, 5), ESA            '画面のキャラクタを読み取って容器に入れる
CLS                                              '画面消去
DIM X(1 TO 10),Y(1 TO 10)         '10個のエサの次の座標位置の容器を作る
DIM HX(1 TO 10),HY(1 TO 10)       '10個のエサの現在の座標位置の容器を作る
FOR I = 1 TO 10                   '10個のエサの最初の座標位置を設定して表示
    X(I) = I * 60
    Y(I) = 0
    PUT (X(I), Y(I)), ESA, XOR
NEXT I
DO                                               '繰り返す
    FOR I = 1 TO 10                              '10個のエサを繰り返す
        ESAMAKI ESA(), X(I), Y(I), HX(I), HY(I)
                                      'エサをまくサブプロシージャを使う
    NEXT I
LOOP
```

```
END
DATA .666.                                    'エサのキャラクタのデータ
DATA 66666
DATA 66666
DATA 66666
DATA .666.

DEFINT E,H,X-Y
SUB ESAMAKI (ESA(), X, Y, HX, HY)             'エサをまくサブプロシージャ
    HX = X: HY = Y                            '現在位置を保存する
    X = X + (RND - .5) * 50                   '新しい横位置の計算
    Y = Y + RND * 50                          '新しい縦位置の計算
    IF Y > 394 THEN                           '画面下の範囲外ならば
        Y = 0                                     '一番上の位置へ
    END IF
    IF (X < 0) OR (X > 634) THEN              '画面の左右の範囲の外ならば
        Y = HY: X = HX                    '保存していた現在位置を次の位置とする
    ELSE                                      '画面の範囲内ならば
        PUT (HX, HY), ESA, XOR                    '現在位置のエサを消し
        PUT (X, Y), ESA, XOR                      '次の位置のエサを描く
    END IF
END SUB
```

前節のプログラムでは、X、Yは次の位置の座標、HX、HYは現在位置の座標の変数でした。そこで、10個のエサを取り扱う場合はこれを10個分用意する必要があります。それが、DIM X(1 TO 10), Y(1 TO 10)とDIM HX(1 TO 10), HY(1 TO 10)です。これは、X(1),X(2)...X(10)というように番号を付けて取り扱うことができるので、FOR....NEXT で Iを1から10まで変化させれば、能率的に仕事ができるのです。

次に、10個それぞれのエサについて、エサを動かすサブプロシージャを使って動作させます。サブプロシージャは独立しているので、データを渡して動作させ、また、処理結果のデータは逆に返してくるようになっています。

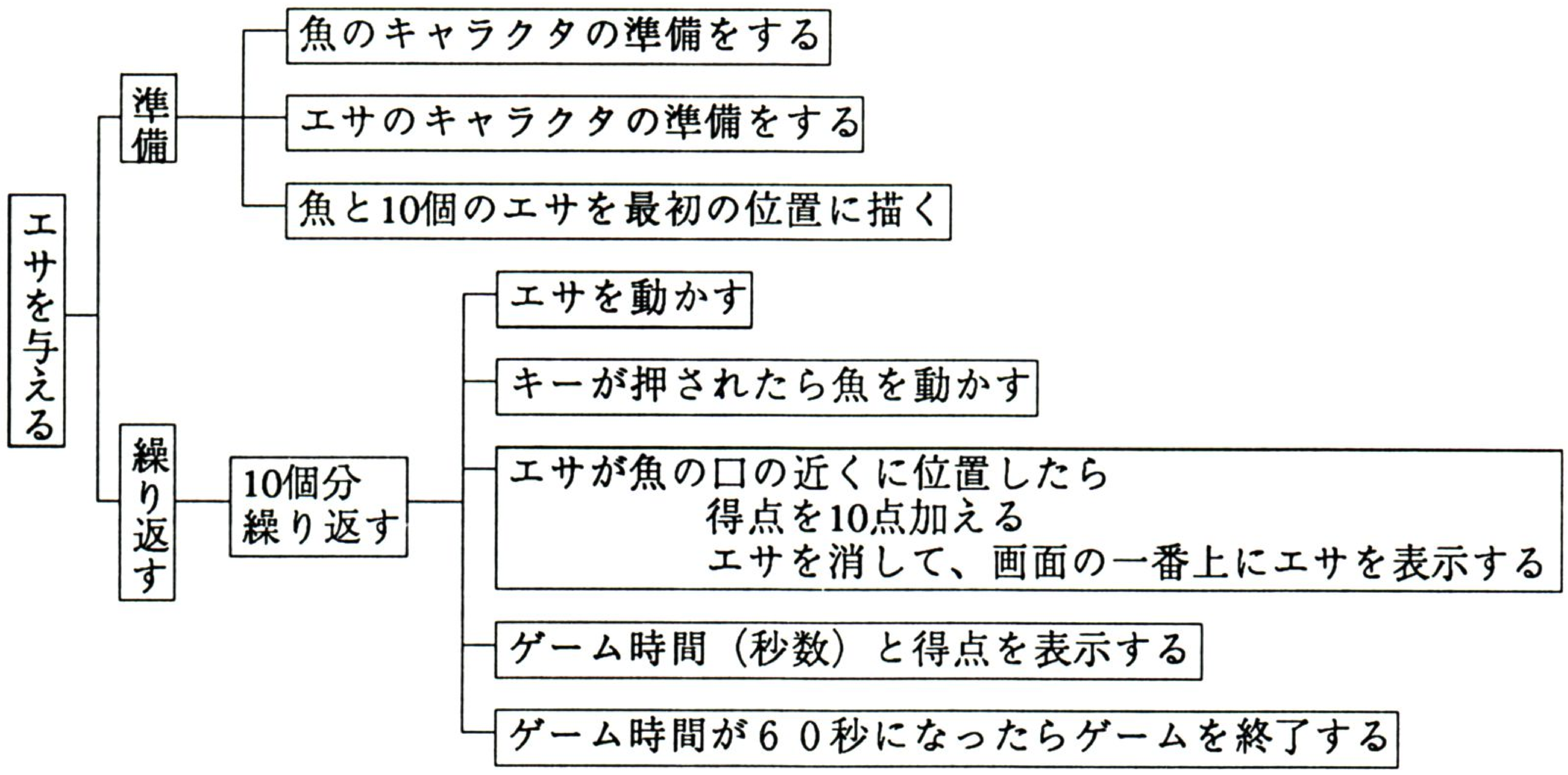
I番目のデータを送って処理させる　　ESAMAKI ESA(), X(I), Y(I), HX(I), HY(I)

I番目のデータを受け取ってエサをまく　　SUB ESAMAKI (ESA(), X, Y, HX, HY)

処理が終ると結果のデータを
I番目のデータとして返す　　ESAMAKI ESA(), X(I), Y(I), HX(I), HY(I)

3．8　ゲームに挑戦

ゲーム作りの材料が揃いました。魚を泳がすプログラムとエサを与えるプログラムを組み合せるとゲームができます。ゲームのプログラムの構造は次のとおりです。




```
DECLARE SUB ESAMAKI (ESA%(), X%, Y%, HX%, HY%)
'アニメ９ 魚がエサを食べるゲーム
DEFINT H-J, M, O, S-U, X-Y                  '整数型変数名を決める
RANDOMIZE TIMER                             '乱数系列を設定する
TI& = TIMER                                 '時間をセットする
TEN = 0                                     '点数を0とする
CLS                                         '画面消去
TATE = 17: YOKO = 48                        'キャラクタの縦横の大きさを決める
FOR Y = 1 TO TATE                   'キャラクタのデータを読んで画面に表示する
    READ A$
    FOR X = 1 TO YOKO
        PSET (X, Y), VAL(MID$(A$, X, 1))                    '左向きの魚を描く
        PSET (YOKO - X + 1, Y + TATE), VAL(MID$(A$, X, 1))
                                                            '右向きの魚を描く
    NEXT X
NEXT Y
OOKISA = (4 + 4 * ((YOKO + 7) ¥ 8) * TATE) ¥ 2
                    '画面から読み取ったキャラクタを入れる容器の大きさを計算する
DIM SAKANA(1 TO OOKISA) AS INTEGER  '左向きのキャラクタを入れる容器を作る
DIM SAKANA2(1 TO OOKISA) AS INTEGER '右向きのキャラクタを入れる容器を作る
GET (1, 1)-(YOKO, TATE), SAKANA     '左向きのキャラクタを画面から読み取る
GET (1, 1 + TATE)-(YOKO, TATE * 2), SAKANA2
                                    '右向きのキャラクタを画面から読み取る
CLS                                         '画面消去
```

```
MUKI$ = "HIDARI"                              '現在の向きをメモしておく
HIDARI = 0                                    '移動範囲の画面の左端
MIGI = 639 - YOKO                             '移動範囲の画面の右端
UE = 0                                        '移動範囲の画面の上端
SITA = 399 - TATE                             '移動範囲の画面の下端
FOR Y = 1 TO 5                                'エサを描く
    READ A$
    FOR X = 1 TO 5
        PSET (X, Y), VAL(MID$(A$, X, 1))
    NEXT X
NEXT Y
OOKISA = (4 + 4 * ((5 + 7) ¥ 8) * 5) ¥ 2
                              'エサのキャラクタのデータの容器の大きさの計算
DIM ESA(1 TO OOKISA) AS INTEGER     'エサのキャラクタのデータの容器を作る
GET (1, 1)-(5, 5), ESA                        'エサのキャラクタを読み取る
CLS                                           '画面消去
DIM X(1 TO 10), Y(1 TO 10)          '10個のエサの次の座標位置の容器を作る
DIM HX(1 TO 10), HY(1 TO 10)      '10個のエサの現在の座標位置の容器を作る
FOR I = 1 TO 10                   '10個のエサの最初の座標位置を決めて表示
    X(I) = I * 60
    Y(I) = 0
    PUT (X(I), Y(I)), ESA, XOR
NEXT I
X = 300: Y = 200                              '最初に魚を表示する位置を決める
PUT (X, Y), SAKANA, XOR                       '最初の位置へ魚を表示する
DO                                            '繰り返す
    FOR I = 1 TO 10                           '10個のエサを繰り返す
        ESAMAKI ESA(), X(I), Y(I), HX(I), HY(I)
                                              'エサをまくサブプロシージャを使う
        K$ = INKEY$                     'キーが押されれば1文字をK$へ入れる
        IF K$ <> "" THEN                      'キーが押されていれば
            HX = X: HY = Y              '魚の現在位置をHX, HYに保管しておく
            S = 20                      '1回に移動するピクセル数をセットする
            SELECT CASE K$
                CASE "4"                      '4が押されていれば
                    X = X - S                     '左へ
                CASE "6"                      '6が押されていれば
                    X = X + S                     '右へ
                CASE "8"                      '8が押されていれば
                    Y = Y - S                     '上へ
                CASE "2"                      '2が押されていれば
                    Y = Y + S                     '下へ
                CASE "7"                      '7が押されていれば
                    X = X - S: Y = Y - S          '左上へ
                CASE "3"                      '3が押されていれば
                    X = X + S: Y = Y + S          '右下へ
                CASE "9"                      '9が押されていれば
                    X = X + S: Y = Y - S          '左上へ
                CASE "1"                      '1が押されていれば
```

```
            X = X - S: Y = Y + S          '右下へ
      END SELECT
      IF (X >= HIDARI) AND (Y >= UE) AND (X <= MIGI) AND
         (Y <= SITA) THEN            '魚の位置が画面の範囲内ならば
          SELECT CASE MUKI$
             CASE "HIDARI"                '左向きならば
                PUT (HX, HY), SAKANA, XOR        '左向きの魚を消す

             CASE "MIGI"                  '右向きならば
                PUT (HX, HY), SAKANA2, XOR       '右向きの魚を消す
          END SELECT
          SELECT CASE X
             CASE IS < HX                 '左へ移動ならば
                MUKI$ = "HIDARI"                  'HIDARIとメモし
                PUT (X, Y), SAKANA, XOR           '新しい位置へ
                                                  '左向きの魚を描く
             CASE IS > HX                 '右へ移動ならば
                MUKI$ = "MIGI"                    'MIGIとメモし
                PUT (X, Y), SAKANA2, XOR          '新しい位置へ
                                                  '右向きの魚を描く
             CASE IS = HX                 '移動しないならば
                SELECT CASE MUKI$
                   CASE "HIDARI"              '現在が左向きならば
                      PUT (X, Y),SAKANA,XOR    '左向きの魚を描き
                   CASE "MIGI"                '現在が右向きならば
                      PUT (X, Y),SAKANA2,XOR   '右向きの魚を描く
                END SELECT
          END SELECT
      ELSE
          X = HX: Y = HY   '魚の位置が画面の範囲外なので範囲内に戻す
      END IF
   END IF
   FOR J = 1 TO 10                      'エサをとったか10個分判定する
      SELECT CASE MUKI$
          CASE "HIDARI"                     '左向きならば
             IF ABS(X(J) - X) < 10 AND
                ABS(Y(J) - Y - 10) < 10 THEN BEEP
                                            '口の近くならば警告音を出す
                PUT (X(J), Y(J)), ESA, XOR         'エサを消す
                Y(J) = 0                   'エサの位置を画面の一番上にして
                PUT (X(J), Y(J)), ESA, XOR         'エサを表示する
                TEN = TEN + 10                     '点数を10点加える
             END IF
          CASE "MIGI"                       '右向きならば
             IF ABS(X(J) - X - YOKO) < 10
                       AND ABS(Y(J) - Y - 10) < 10 THEN BEEP
                PUT (X(J), Y(J)), ESA, XOR
                Y(J) = 0
                PUT (X(J), Y(J)), ESA, XOR
```

```
                    TEN = TEN + 10
                END IF
            END SELECT
        NEXT J
        LOCATE 2, 10
        PRINT "  時間：";TIMER - TI&;"  得点：";TEN;
                                                '時間と得点を表示する
        IF TIMER - TI& = 60 THEN                '60秒経過したら
            LOCATE 5, 10: PRINT "ゲームオーバー"       '終了を表示して
            EXIT DO                                    '終了する
        END IF
    NEXT I
LOOP
END
        (魚とエサのキャラクタのデータは省略します)
        (エサを動かすサブプロシージャは省略します)
```

時間はTIMERでパソコンから取得します。TIMERは0時0分からの秒数です。数が大きいので倍長整数のTI&を使いました。エサを食べた判定は、左向きでは(X,Y+10)、右向きでは(X+YOKO,Y+10)を中心に10ピクセルの範囲内にエサあれば食べたことにします。ABS()は絶対値の関数で、魚の位置とエサの位置の差がマイナスの場合はプラスに変えて、差が10未満であるかどうかを判定しました。

3．9　タイトル画面をつくる

お魚のゲームがなんとかできました。しかし、まだ表紙はありませんしキャラクタをつくる仕掛けも見えるので、これを改善することにしましょう。

まず、次のプログラムを最初の部分のDEFINTの次の行に挿入してください。

```
'QB3_9_1　お魚ゲーム
CLS                                        '画面消去
FOR I = 1 TO 120 STEP 5                    'タイトル画面を描く
    LINE (I, I)-(639 - I, 399 - I), I, B
NEXT I
LOCATE 13, 35: PRINT "お魚ゲーム"          'タイトルを書く
FOR I = 1 TO 50
    FOR J = 1 TO 7
        PALETTE J, ((J + I) MOD 7) + 1     '色をいろいろ変える
        FOR JJ = 1 TO 100
        NEXT JJ
    NEXT J
NEXT I
    PALETTE                                '色をもとに戻す
SCREEN 0, 0, 1, 0
  '表画面にタイトル画面を表示しながら、裏画面に描くモードにする
```

さらに、エサのキャラクタを読み取るステートメント（GET(1,1)-(5,5),ESA）と画面消去（CLS）の間に次のステートメントを挿入してください。

```
SCREEN 0, 0, 0, 0
```

こうして実行すれば、キャラクタをつくる工程が影に隠れてしまい、代わりにタイトル画面が表示されます。

PALETTEは、色を取り替えるステートメントです。

```
PALETTE　変更の対象とする色番号，新たな色番号
```

例えば、LINE(20, 30)-(200, 100), 2, B とすれば、緑色で描きますが、 次に、PALETTE 2, 4　とすれば、すべての緑色が一瞬で赤色に変化します。そしてこれ以後は、色番号2は赤色になるのです。すなわち、プログラムの中の色を一度に取り替えることができるのです。

なお、元に戻すにはPALETTE とします。これで色番号2は緑色に戻ります。
タイトル画面では、このPALETTEの色を変える機能を使って、画面の色を変えたのです。

SCREEN は、プログラム内で操作する画面とディスプレイに表示する画面を指定するステートメントです。
実は、PC9800はオモテの画面とウラの画面の2枚の画面をもっています。今まではオモテの画面のみを使っていましたが、今回はオモテの画面にタイトルを表示しておいて、その間にウラの画面を使って魚とエサのキャラクタを作るようにしました。
SCREEN 0, 0, 1, 0 を実行すると、オモテの画面を表示しながら、ウラの画面を使ってキャラクタなどを読み書きできます。また、SCREEN 0,0,0,0 とすると、オモテ画面を表示しながら、オモテ画面を操作します。
マニュアルの「SCREEN　スクリーンモード，パレットスイッチ，コマンドの対象となる画面，表示する画面」という説明ではわかりにくいのですが、実際に使う例は次のとおりです。

オモテ画面を操作し、オモテ画面を表示する→	SCREEN 0, 0, 0, 0
オモテ画面を操作し、ウラ画面を表示する→	SCREEN 0, 0, 0, 1
ウラ画面を操作し、オモテ画面を表示する→	SCREEN 0, 0, 1, 0
ウラ画面を操作し、ウラ画面を表示する→	SCREEN 0, 0, 1, 1

３．１０　オルゴールのＢＧＭ

Ｑｕｉｃｋ　ＢＡＳＩＣは、オルゴールの演奏もできます。それも、バックグラウンドミュージック（ＢＧＭ）ができるので、画面では、アニメやイラストを表示しながら、演奏できるわけです。

では、基本的なオルゴールの鳴らし方から、ＢＧＭまで、順序に説明します。

（１）音階

ドレミファソラシドは、CDEFGAB>Cと表します。

例　　PLAY "CDEFGAB>C"

>は１オクターブ上がることです。

例　　PLAY "CDEFGAB>CC<BAGFEDC"

<は１オクターブ下がることです。

（２）オクターブ

一番低いオクターブをO0（オーゼロ）と表し、一番高いオクターブをO6（オーロク）と表します。

例　　PLAY "O0 CDEFGAB>C"
　　　PLAY "O6 CDEFGAB"
　　　PLAY "O3 CDEFGAB>C"

（３）音符の長さ

２分音符をL2、４分音符をL4、８分音符をL8と表します。６４分音符まで可能です。

例　　PLAY "O3 L4 CDEFGAB>C"
　　　PLAY "O3 L8 CDEFGAB>C L16 CDEFGAB>C"

（４）半音

半音上げは ＋ と表し、半音下げは － と表します。

例　　PLAY "O3 L8 GAB>CDEF+G"

（５）個々の音符の長さと符点音符

L4は、４分音符ですが、全部が４分音符では音楽にはなりません。そこで、４分音符の多い音楽では、はじめにL4として、長さを書かない場合には４分音符とし、４分音符以外の場合は長さを書きます。

また、点を付して符点音符を表します。

例　　PLAY "O3 L4 C.D8 E.F8 G.A8"
　　　C.は４分音符ド符点，D8は８分音符レ，ということです。
　　　PLAY "O3 L8 C4.D E4.F G4.A"
　　　C4.は４分音符ド符点，D8は８分音符レ，ということです。
　　　どちらも同じ音符ですが、L4とL8の違いでその後の書き方が変わり。
　　　PLAY "O3 L8 G4.FEG>CDE2C2"

（6）速さ

１分当りの４分音符の数で表します。

例　PLAY "T32 O3 L8 G4.FEG>CDE2C2"

T32は１分間に４分音符が32ということです。一番遅いテンポ**が32です。**

PLAY "T120 O3 L8 G4.FEG>CDE2C2"

初期値の速さが120です。

PLAY "T255 O3 L8 G4.FEG>CDE2C2"

一番速いテンポが255です。

（7）音符の演奏する長さ
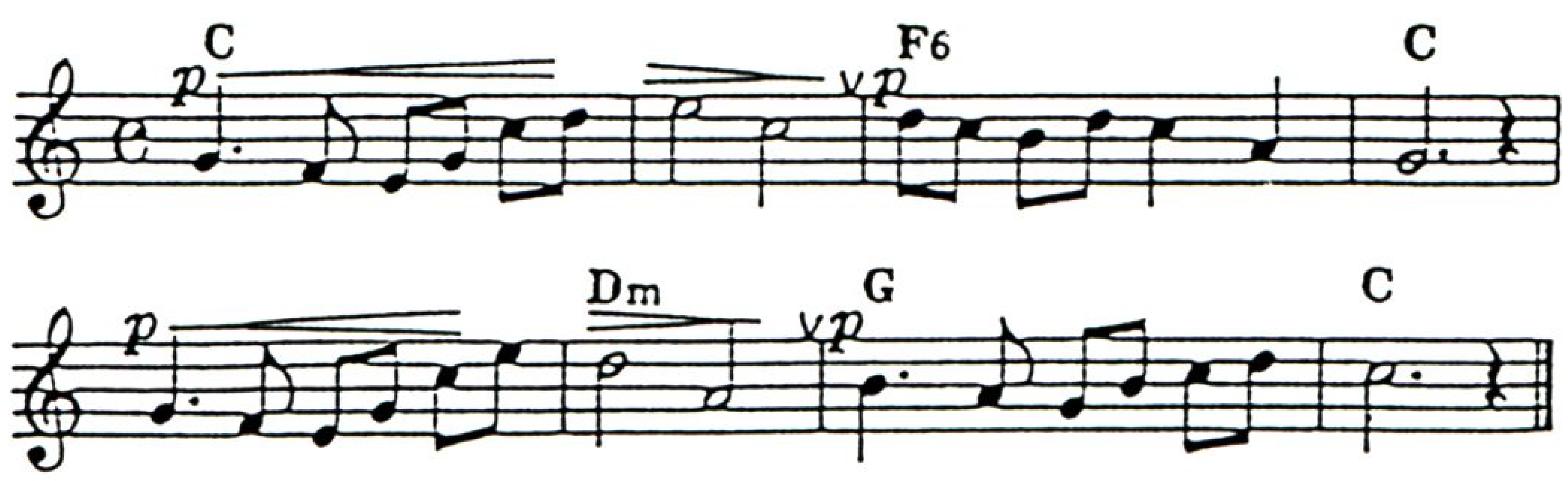
MSが４分の３の長さ、MNが８分の７の長さ、MLが完全な長さで演奏します。

そこで、MSはスタッカートのように演奏し（スタッカートではない）、MLではスラーで演奏します。普通は、MNでよいようです。また、MLで同音ではタイになります。

例　PLAY "T120 O3 L8 ML G4.FEG>CDE2C2"
　　PLAY "T120 O3 L8 MN G4.FEG>CDE2C2"
　　PLAY "T120 O3 L8 MS G4.FEG>CDE2C2"

実際の曲のプログラム例は次のとおりです。

燃えろよ燃えろ　　フランス民謡



＜燃えろよ燃えろ＞
（作者不詳）
燃えろよ燃えろ
炎よ燃えろ
火の粉を巻きあげ
天までこがせ

```
PLAY "T120 O3 L8 MN G4.FEG>CDE2C2"
PLAY "DC<B>DC4<A4G2.P4"
PLAY "G4.FEG>CED2<A2"
PLAY "B4.AGB>CDC2.P4"
```

さて、いよいよＢＧＭのプログラミングです。

```
ON PLAY(5) GOSUB ONGAKU '←ＢＧＭ音符が少ないとき(5)ONGAKU:のルーチン
PLAY ON                 '←上記の設定を実行可能にする　　　を実行するよう設定
GOSUB ONGAKU:           '←ONGAKU:ルーチンを実行
CLS
DO
    FOR XX = 1 TO 640 STEP 47
```

```
            FOR X = 1  TO 640 STEP 47
                FOR Y = 1  TO 400 STEP 47
                    FOR R = 1  TO 400 STEP 40
                        LINE (XX, 0)-(X,  Y), XX
                    NEXT R
                NEXT Y
            NEXT X
    NEXT XX
LOOP
END
ONGAKU:                              '←ONGAKU:ルーチン
    READ  M$                         '←ONGAKUDATAを読む
    IF  M$ = ""  THEN                '←データがなければ
        RESTORE ONGAKUDATA:            '←データをはじめから
        READ  M$                       '←読み直す
    END IF
    PLAY  M$                         '←データを演奏する
RETURN                               '←戻る
ONGAKUDATA:                          '←音楽データ
DATA "MB T120 O3 L8 MN F.F16 F.G16 A.A16 A.A16 G.F16 G.A16 F.F16 C4"
DATA "A.A16 A.B-16 > C.C16 C.C16 < B-.A16 B-.>C16 < A4. > C8"
DATA "<A4.>C < A4.>C < A.>C16 < A.>C16 <A2"
DATA ""
```
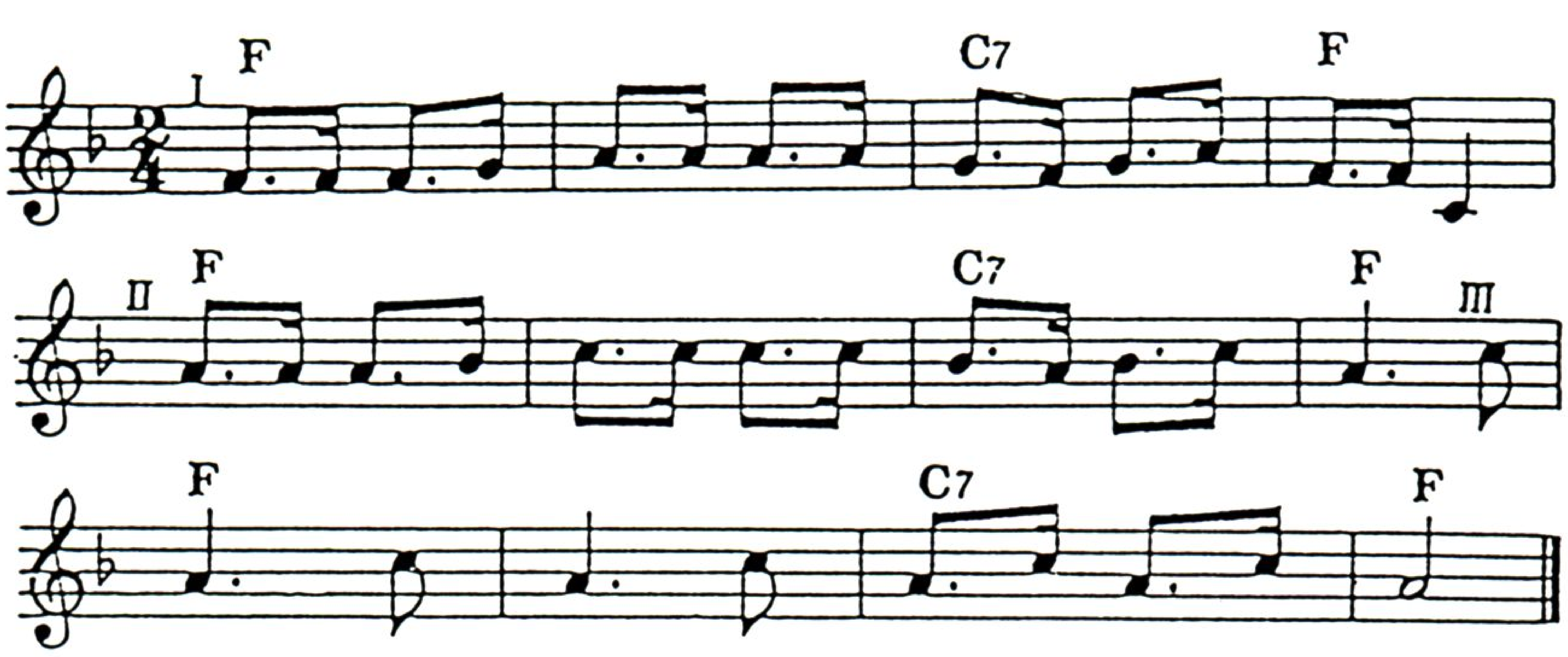
ON PLAY(5) GOSUB ONGAKU: は、楽譜のデータがなくなって、ＢＧＭが停止しないようにしている仕掛です。 これによって、何回も演奏を繰り返すことができるのです。ただし、これは、機能の設定だけで、次の PLAY ON がこの機能をＯＮにしています。

DATA "MB T120 ‥‥ の MB がＢＧＭとなるように指示しています。

ＢＧＭは、アニメやイラストを楽しくする効果があります。

静 か な 湖 畔

スイス民謡



３．１１　効果音でおもしろく

パソコンのBEEP音は単純な音ですが、この装置を利用して、いろいろな周波数を与えることによって、効果音を発生することができます。

```
SOUND 1000, 7
```

この１行のプログラムを実行すると1000Hz（ヘルツ）の音が0.385秒間鳴ります。時間は、1が0.055秒ですから、7ならば7×0.055＝0.385秒ということです。

```
SOUND 周波数， 時間
```

周波数は38Hzから32767Hzまで使えますが、人間の耳に聴こえるのは10000Hzぐらいまでのようです。

```
'QB3_11_1 救急車のサイレン
DO
    SOUND 1000, 7
    SOUND 330, 7
LOOP
```

```
'QB3_11_2 何かが落下する効果音
FOR I = 3000 TO 300 STEP -10
    SOUND I, .2
NEXT
```

```
'QB3_11_3 何かが上昇する効果音
FOR I = 300 TO 3000 STEP 10
    SOUND I, .2
NEXT
```

```
'QB3_11_4 何かが弾む効果音
FOR I = 1 TO 20
    SOUND 300, .2
    SOUND 32767, 10 / I
NEXT
```

第❹章

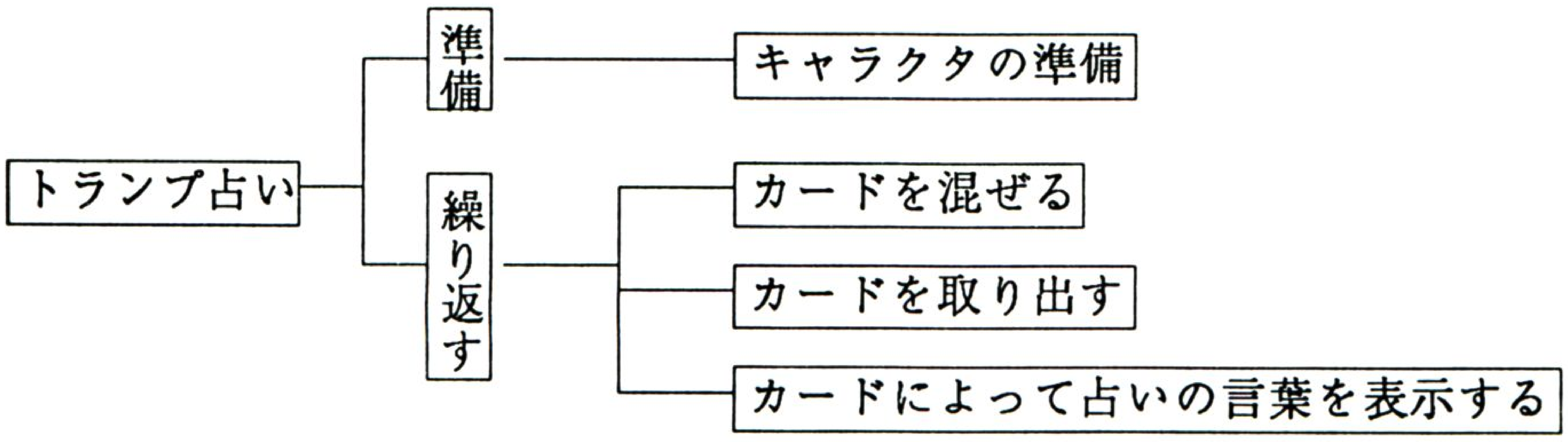
トランプ占い

キャラクタのフロッピーディスクへの保存とレコード型変数の使い方を学びます。

4．1　キャラクタの保存

トランプ占いのプログラムを作りましょう。この章では、フロッピーディスクとレコード型の変数の使い方などを学びます。

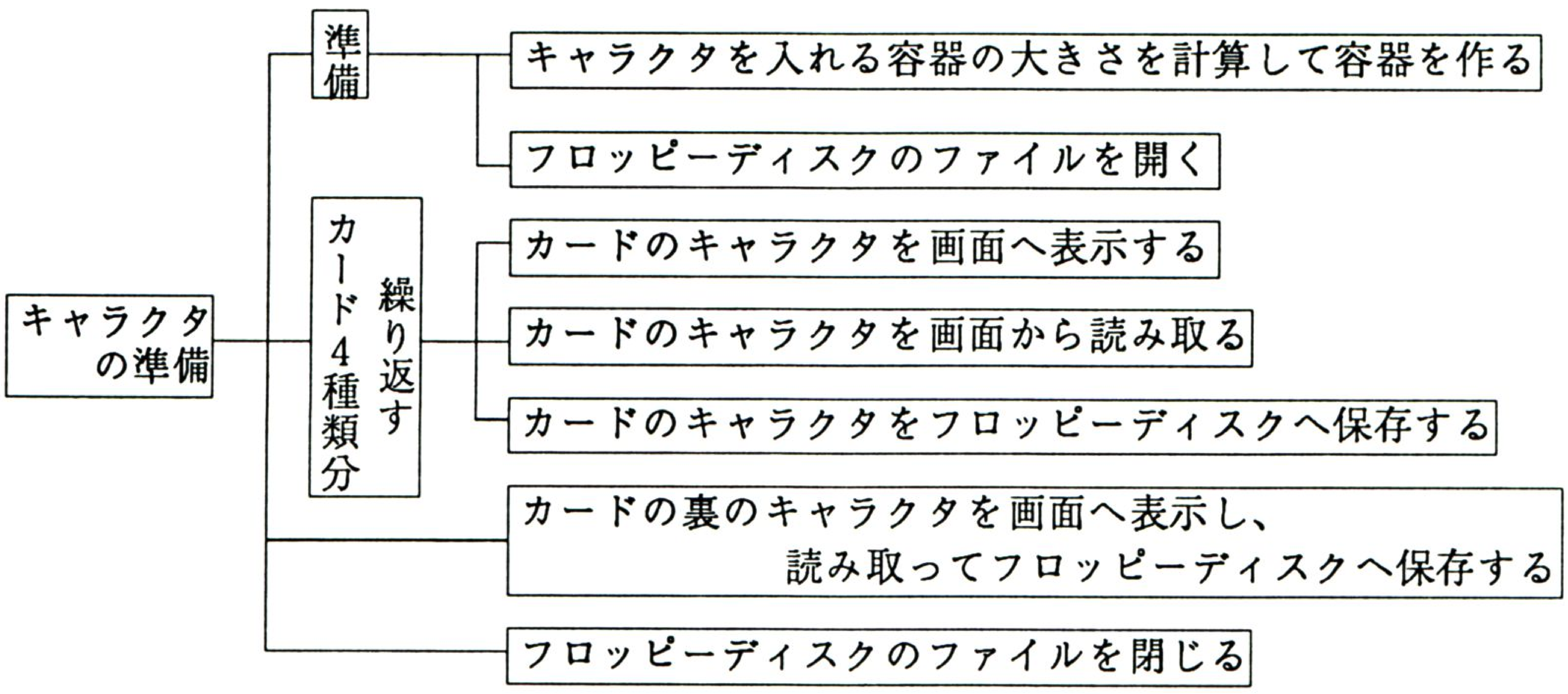
トランプ占いのプログラム全体の構造の概要は次のとおりです。



まず、カードのキャラクタを作ります。

カードのキャラクタは、ダイヤ、ハート、スペード、クラブの4種類及びカードの裏面です。今回のプログラムでは裏面は使わないのですが、別のトランプゲームを作成するときのために加えておきました。4種類のキャラクタの作り方は同じなので、この処理をサブプロシージャへまとめました。

プログラムの構造は次のとおりです。




```
DECLARE SUB HOZON (ZUKEI%(), I1%, I2%)
'占い１      ****トランプ占い　カードをつくり保存する****
DEFINT B, I, N, T, Y                          '整数型変数
CLS                                           '画面消去
B = (4 + 4 * ((32 + 7) ¥ 8) * 64) ¥ 2
                                  'トランプの絵を入れる容器の大きさを計算する
DIM DIAMOND(1 TO B) AS INTEGER                'ダイヤの容器を作る（整数）
DIM HEART(1 TO B)   AS INTEGER                'ハートの容器を作る
DIM SPADE(1 TO B)   AS INTEGER                'スペードの容器を作る
DIM CLUB(1 TO B)    AS INTEGER                'クラブの容器を作る
```

```
DIM URA(1 TO B)        AS INTEGER                    'カードの裏面の容器を作る
LINE (1, 1)-(32, 8), 7, BF                           'トランプの絵を描く
LINE (1, 33)-(32, 64), 7, BF
OPEN "B:CARD.DAT" FOR RANDOM AS #1 LEN = LEN(DIAMOND)     'ファイルを開く
HOZON DIAMOND(), 1, B        '１からＢまでのレコード番号でダイヤの絵を保存
HOZON HEART(), B + 1, B * 2
                          'Ｂ＋１からＢ×２までのレコード番号でハートの絵を保存
HOZON SPADE(), B * 2 + 1, B * 3
                     'Ｂ×２＋１からＢ×３までのレコード番号でスペードの絵を保存
HOZON CLUB(), B * 3 + 1, B * 4
                      'Ｂ×３＋１からＢ×４までのレコード番号でクラブの絵を保存
FOR N = 1 TO 32                                      'カードの裏面を描く
    LINE (N, 1)-(N, 64), N
NEXT N
FOR N = 1 TO 64 STEP 2
    LINE (1, N)-(32, N), N
NEXT N
GET (1, 1)-(32, 64), URA                             '裏面のキャラクタを読む
FOR I = B * 4 + 1 TO B * 5
    PUT #1, I, URA(I - B * 4)  '裏のキャラクタをフロッピーディスクへ保存
NEXT I
CLOSE #1                                             'ファイルを閉じる
            '↓テストのため描いてみる
PUT (1, 1), DIAMOND, PSET                            'ダイヤを描く
PUT (41, 1), HEART, PSET                             'ハートを描く
PUT (81, 1), SPADE, PSET                             'スペードを描く
PUT (121, 1), CLUB, PSET                             'クラブを描く
PUT (161, 1), URA, PSET                              '裏を描く
END
DATA 77777777777777747777777777777777
DATA 77777777777777444777777777777777
DATA 77777777777774444477777777777777
DATA 77777777777744444447777777777777
DATA 77777777777444444444777777777777
DATA 77777777774444444444477777777777
DATA 77777777744444444444447777777777
DATA 77777777444444444444444777777777
DATA 77777774444444444444444477777777
DATA 77777744444444444444444447777777
DATA 77777444444444444444444444777777
DATA 77774444444444444444444444477777
DATA 77777444444444444444444444777777
DATA 77777744444444444444444447777777
DATA 77777774444444444444444477777777
DATA 77777777444444444444444777777777
DATA 77777777744444444444447777777777
DATA 77777777774444444444477777777777
DATA 77777777777444444444777777777777
DATA 77777777777744444447777777777777
DATA 77777777777774444477777777777777
DATA 77777777777777444777777777777777
DATA 77777777777777747777777777777777
DATA 77777777777777777777777777777777
```

```
DATA 77777777744477777777744477777777
DATA 77777774444444777774444444777777
DATA 77777744444444777744444444777777
DATA 77777444444444477444444444477777
DATA 77777444444444477444444444447777
DATA 77774444444444477444444444447777
DATA 77774444444444444444444444447777
DATA 77774444444444444444444444447777
DATA 77774444444444444444444444447777
DATA 77774444444444444444444444447777
DATA 77777444444444444444444444477777
DATA 77777444444444444444444444477777
DATA 77777444444444444444444444777777
DATA 77777744444444444444444447777777
DATA 77777774444444444444444477777777
DATA 77777777444444444444444777777777
DATA 77777777774444444444477777777777
DATA 77777777777444444444777777777777
DATA 77777777777744444447777777777777
DATA 77777777777774444777777777777777
DATA 77777777777774444777777777777777
DATA 77777777777777444777777777777777
DATA 77777777777777747777777777777777
DATA 77777777777777777777777777777777

DATA 77777777777777077777777777777777
DATA 77777777777777000777777777777777
DATA 77777777777777000777777777777777
DATA 77777777777770000077777777777777
DATA 77777777777700000007777777777777
DATA 77777777777000000000777777777777
DATA 77777777777000000000077777777777
DATA 77777777000000000000000777777777
DATA 77777700000000000000000007777777
DATA 77777000000000000000000000777777
DATA 77770000000000000000000000077777
DATA 77770000000000000000000000077777
DATA 77770000000000000000000000077777
DATA 77770000000000000000000000077777
DATA 77770000000000000000000000077777
DATA 77770000000000000000000000077777
DATA 77777000000000000000000000777777
DATA 77777000000007707700000007777777
DATA 77777770000077000770000077777777
DATA 77777777777770000077777777777777
DATA 77777777777700000007777777777777
DATA 77777777777000000000777777777777
DATA 77777777770000000000077777777777
DATA 77777777777777777777777777777777
DATA 77777777777770000077777777777777
DATA 77777777777000000000077777777777
DATA 77777777770000000000007777777777
DATA 77777777700000000000000777777777
DATA 77777777700000000000000777777777
DATA 77777777700000000000000777777777
DATA 77777777770000000000007777777777
DATA 77777777770000000000007777777777
```

```
DATA 77777700000700000000700000777777
DATA 77777000000007000070000000077777
DATA 77770000000000000000000000007777
DATA 77770000000000000000000000007777
DATA 77770000000000000000000000007777
DATA 77770000000000000000000000007777
DATA 77770000000000000000000000007777
DATA 77770000000000000000000000077777
DATA 77777000000000000000000000077777
DATA 77777770000077707770000077777777
DATA 77777777777777000777777777777777
DATA 77777777777770000077777777777777
DATA 77777777777700000007777777777777
DATA 77777777777000000000777777777777
DATA 77777777770000000000077777777777
DATA 77777777777777777777777777777777
DEFINT I,X-Z
SUB HOZON (ZUKEI(), I1, I2)                  'サブプロシージャのHOZON
    FOR Y = 9 TO 32
        READ A$                              '図形のデータを読む
        FOR X = 1 TO 32
            PSET (X, Y), VAL(MID$(A$, X, 1)) '画面にトランプを表示する
        NEXT X
    NEXT Y
    GET (1, 1)-(32, 64), ZUKEI
    FOR I = I1 TO I2
        PUT #1, I, ZUKEI(I - I1 + 1)         '絵をディスクへ保存する
    NEXT I
END SUB
```

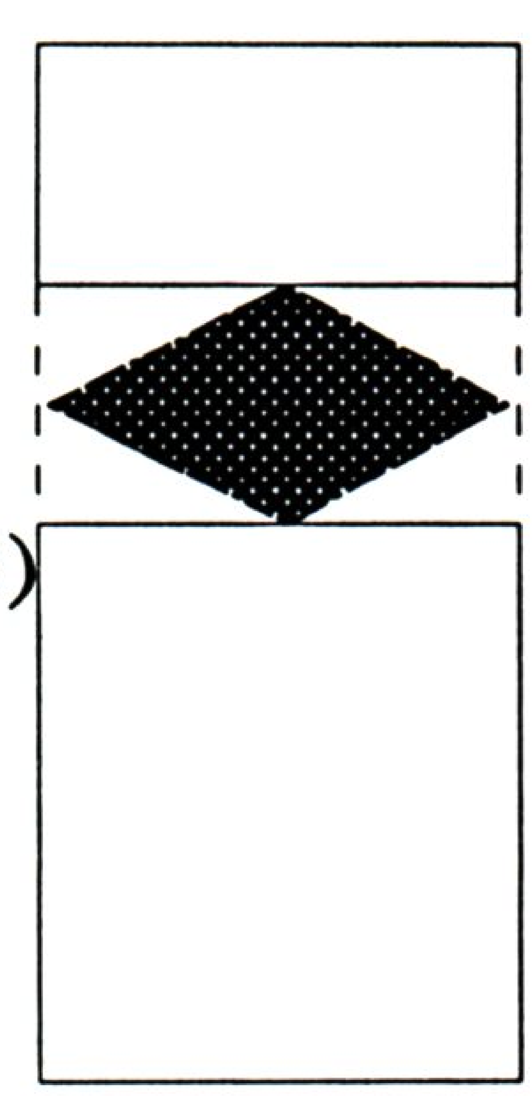
DIM DIAMOND(1 TO B) AS INTEGER は、ダイヤのキャラクタを入れる容器です。

AS INTEGER は整数としてということで、２バイト（１６ビット相当）の領域をB個使用することです。

LINE (1, 1)-(32, 8), 7, BF 及び LINE (1, 33)-(32, 64), 7, BF は、カードの白い部分をあらかじめ描いたのです。

サブプロシージャHOZONの中では、(1,9)-(32,32) の部分に図を描くようにしました。

(1,1)

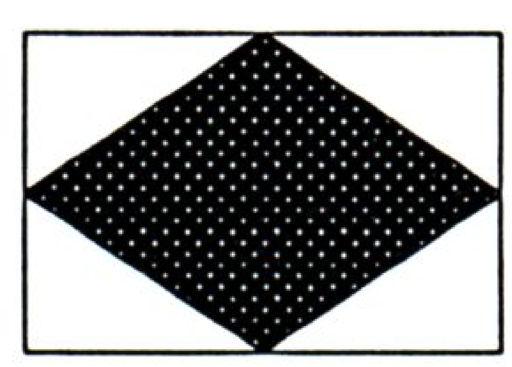


(32,8)

(1,33)

(32,64)

(1,9)



(32,32)

メインプロシージャからサブプロシージャへのデータの受渡しは、次のようにします。

HOZON CLUB(), B * 3 + 1,B * 4

SUB HOZON (ZUKEI(),I1,I2)

OPEN "B:CARD.DAT" AS RANDOM AS #1 LEN=LEN(DIAMOND)は、これから使うファイルの指定です。

OPENはファイルを開くことです。仕事の終りにはファイルを閉じます(CLOSE)。

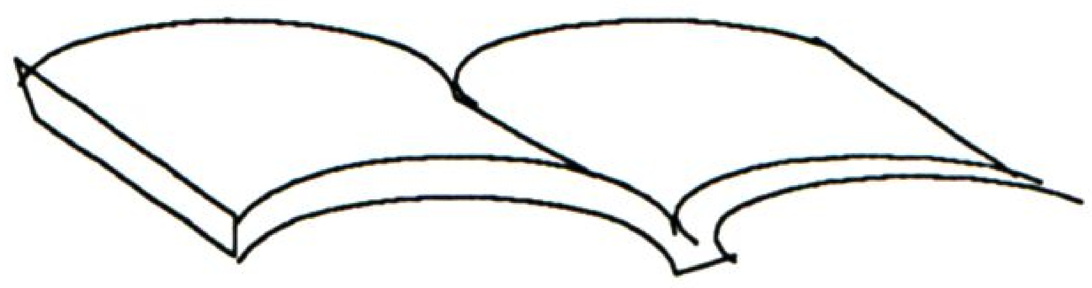
FOR RANDOMは、ランダムアクセスファイルであることを示しています。ランダムアクセスファイルは、ファイルのどの位置でも読み書きができるようなファイルの形式をいいます。そのために、入力するデータの位置（レコード番号＝要するに何番目のデータであるか）を書く必要があります。

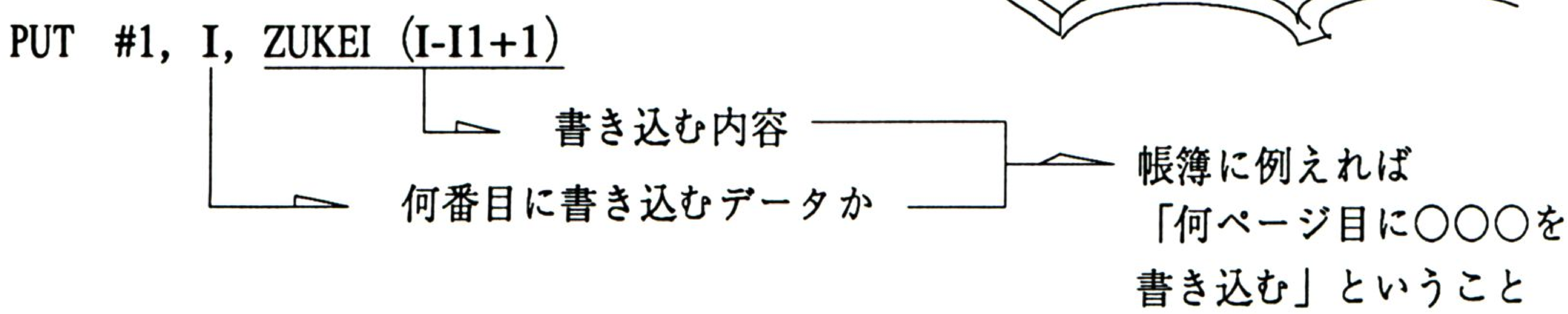
AS #1は、ファイルに#1という番号を付けたのです。入力や出力するときは、この番号を使います。

LEN=LEN(DIAMOND)は１レコード（１件分のデータ）の長さを指示しています。右辺のLENは長さを計算するステートメントです。

PUT #1, I, ZUKEI(I-I1+1)という形式で書き込みます。ここでは、Iがレコード番号でZUKEI(I-I1+1)がデータの内容です。

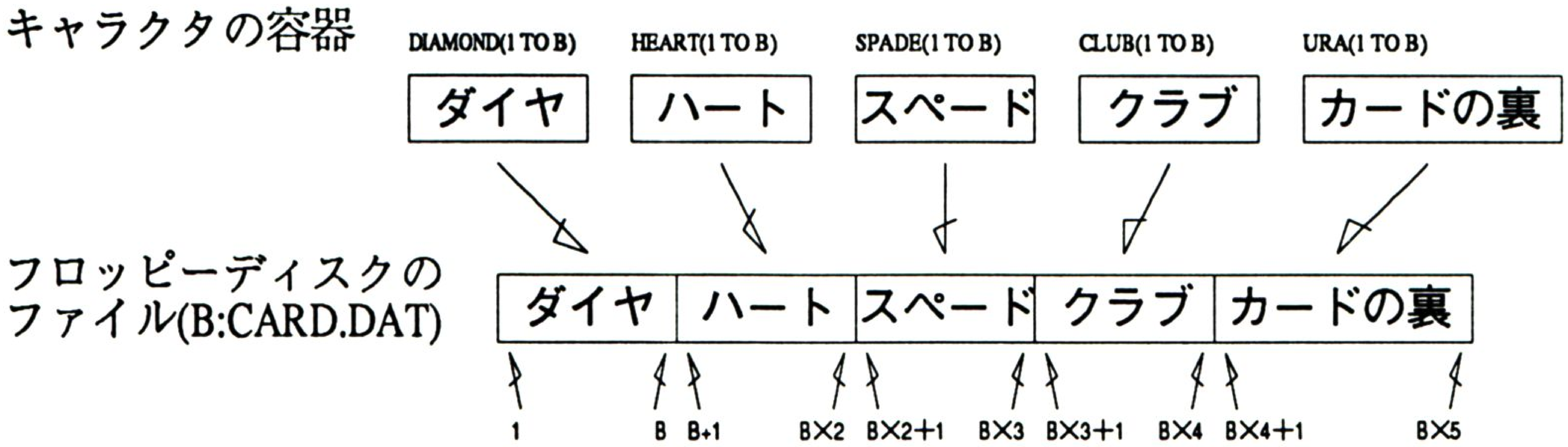
OPENは帳簿を開くようなもの





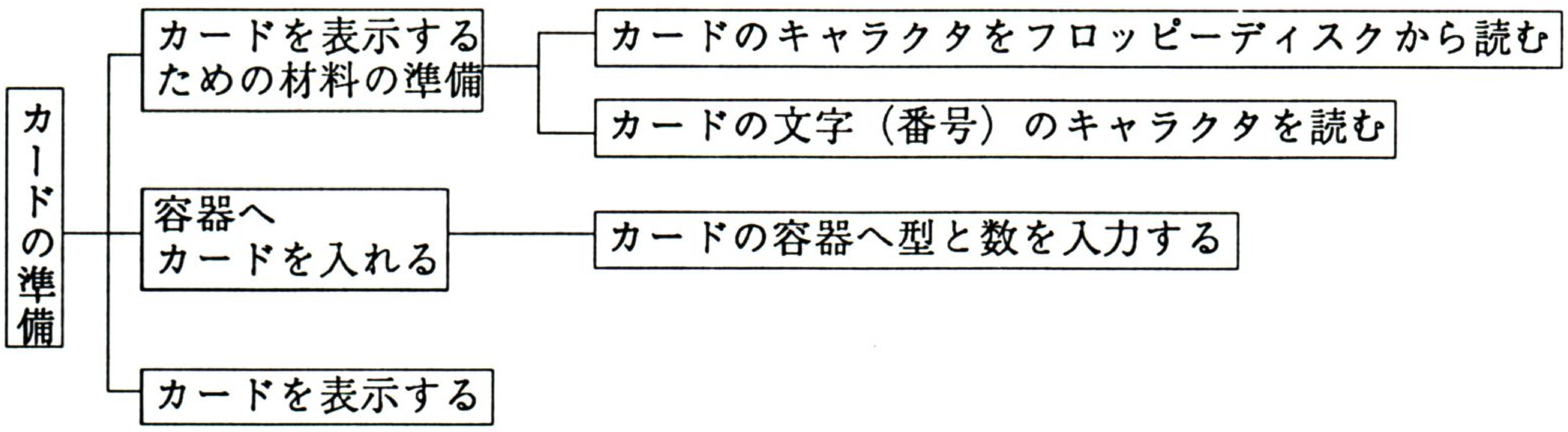
一つのファイルに５種類のキャラクタを入力したので、フロッピーディスクでの入力位置に気を使う必要があります。

そこで、ダイヤは1からBまで、ハートはB+1からB×2までというように複雑になってしまいました。



４．２　カードの準備

　カードを表示するためのキャラクタをつくってフロッピーディスクに保存したので、次に、カードをパソコン内に用意してそれを画面へ表示します。
　プログラムの構造は次のとおりです。




```
DECLARE SUB CARDHYOUZI (CARD AS ANY, MX AS INTEGER, MY AS INTEGER)
DECLARE SUB CARDYOMI (CARDGR() AS INTEGER, A AS INTEGER, B AS INTEGER)
'占い２　　****トランプ占い　カードを用意する****
DEFINT A-B, I-K, T, X-Y
CLS                                                    '←画面消去
TATE = 64: YOKO = 32                                   '←トランプの大きさ
B = (4 + 4 * ((YOKO + 7) ¥ 8) * TATE) ¥ 2
                                   '←トランプの絵を入れる容器の大きさを計算する
DIM DIAMOND(1 TO B) AS INTEGER                         '←ダイヤの容器を作る
DIM HEART(1 TO B)   AS INTEGER                         '←ハートの容器を作る
DIM SPADE(1 TO B)   AS INTEGER                         '←スペードの容器を作る
DIM CLUB(1 TO B)    AS INTEGER                         '←クラブの容器を作る
DIM URA(1 TO B)     AS INTEGER                         '←カードの裏面の容器を作る
DIM MOJI(1 TO 13)   AS STRING * 2                      '←カードの文字の容器を作る

TYPE CARDTYPE                                          '←カード型のデータを決める
    KATA AS INTEGER                                       '←型は整数
    SUU  AS INTEGER                                       '←数は整数
END TYPE                                                  '←印をつける
DIM CARD(1 TO 52) AS CARDTYPE                        '←カード５２枚分の容器を作る

OPEN "B:CARD.DAT" FOR RANDOM AS #1 LEN = LEN(DIAMOND)     '←ファイルを開く
    CARDYOMI DIAMOND(), 1, B                         '←カードの絵のデータを読み取る
    CARDYOMI HEART(), B + 1, B * 2
    CARDYOMI SPADE(), B * 2 + 1, B * 3
    CARDYOMI CLUB(), B * 3 + 1, B * 4
    CARDYOMI URA(), B * 4 + 1, B * 5
CLOSE #1                                               '←ファイルを閉じる
FOR I = 1 TO 13
    READ MOJI(I)                                       '←カード用の文字を読み取る
NEXT I
```

```
                                        '←カード箱の中にカードを入れる
K = 0
FOR I = 1 TO 4                          '←カードは４種類ある
    FOR J = 1 TO 13                     '←１種類のカードは１３枚ある
        K = K + 1                       '←通して何番目のカードか数える
        CARD(K).KATA = I                '←型に１から４の数を入れる
        CARD(K).SUU = J                 '←数に１から１３の数を入れる
    NEXT J
NEXT I
                                        '←カード箱のカードを順に表示する
K = 0
FOR Y = 1 TO 16 STEP 5                  '←表示位置の行数を決める
    FOR X = 1 TO 61 STEP 5              '←表示位置の列数を決める
        K = K + 1                       '←通して何番目にあるか計算する
        CARDHYOUZI CARD(K), X, Y        '←決めた位置へカードを表示する
    NEXT X
NEXT Y
END
DATA A , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 ,10, J , Q , K   '←カードの数の文字

DEFSNG A-B, I-K, T, X-Y
SUB CARDHYOUZI (CARD AS CARDTYPE, MX AS INTEGER, MY AS INTEGER)
                                        '←カードの表示
SHARED DIAMOND() AS INTEGER, HEART() AS INTEGER
SHARED SPADE() AS INTEGER, CLUB() AS INTEGER, URA() AS INTEGER
SHARED MOJI() AS STRING * 2
                                        '←メインとデータを共有する
DEFINT L, X-Y
COLOR 1                                 '←文字の色を青にする
LX = MX + 2                             '←画面中のカードの文字の
LY = MY + 3                               '位置の計算
LOCATE LY, LX                           '←カーソルを文字の位置へ移動
PRINT MOJI(CARD.SUU);                   '←文字を書く
X = MX * 8                              '←グラフィック画面中の
Y = MY * 16                               'カードの位置の計算
SELECT CASE CARD.KATA                   '←カードの絵を描く
    CASE 1                                  '←１ならば
        PUT (X, Y), DIAMOND, XOR                '←ダイヤを描く
    CASE 2                                  '←２ならば
        PUT (X, Y), HEART, XOR                  '←ハートを描く
    CASE 3                                  '←３ならば
        PUT (X, Y), SPADE, XOR                  '←スペードを描く
    CASE 4                                  '←４ならば
        PUT (X, Y), CLUB, XOR                   '←クラブを描く
    CASE 5                                  '←５ならば
        PUT (X, Y), URA, XOR                    '←裏を描く
END SELECT
COLOR 7                                 '←色を白色にする
END SUB
```

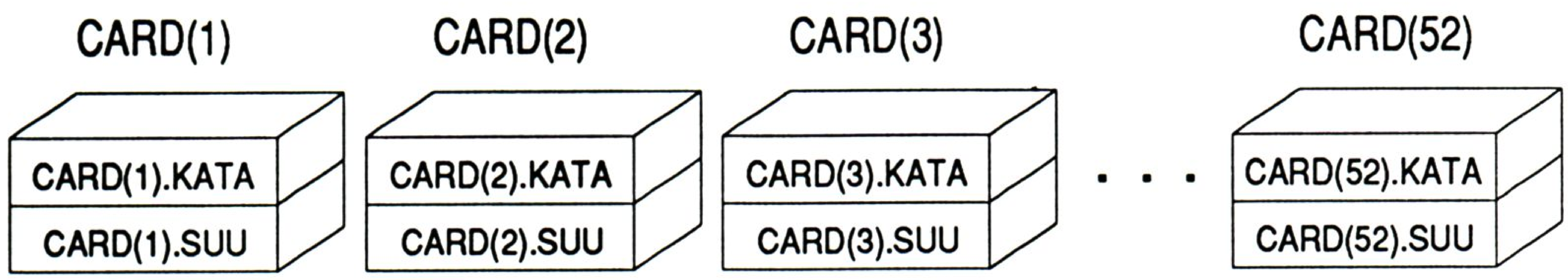
```
DEFSNG L, X-Y
SUB CARDYOMI (CARDGR() AS INTEGER, A AS INTEGER, B AS INTEGER)
DEFINT I-J                                   '←絵のデータをフロッピーから読む
J = 0
FOR I = A TO B                               '←1からB番までのレコード番号で
    J = J + 1                                    '←レコード番号を計算する
    GET #1, I, CARDGR(J)                         '←絵のデータを読み取る
NEXT I
END SUB
```

カードを準備するということは、カードを入れる容器を用意してカードの型と数を入力しておくことです。１枚のカードは、型と数の二つのデータをもっているので、次のように書くことができます。

```
TYPE CARDTYPE              ←CARDTYPEという形状の容器を作る
    KATA AS INTEGER            ←型は整数
    SUU  AS INTEGER            ←数は整数
END TYPE
```

そして、カードの枚数は１３×４＝５２枚ですから、次のようにカードの容器を５２個用意しました。

```
DIM CARD(1 TO 52) AS CARDTYPE
```



容器の形状は、これまでは整数型のように文法上すでに定められている型指定だけでしたが、CARDという容器の形状は、TYPE....END TYPEというステートメントによってプログラマーが自らデータ構造を定めたCARDTYPEという指定になっています。

このような容器のことを、レコード型の変数といいます。この機能は、Ｃ言語やＰＡＳＣＡＬにはありましたが、ＢＡＳＩＣにはありませんでした。

このレコード型の変数指定によって、たとえば18枚目のカードはCARD(18)として取り扱うと同時に、18枚目のカードの型はCARD(18).KATA、18枚目のカードの数はCARD(18).SUUとして取り扱うことができるのです。

KATA及びSUUはともに整数型の指定ですが、この内容を次のように決めます。

○カードの型　ダイヤ：1　ハート：2　スペード：3　クラブ：4

○カードの数　Ａ：1　２：2　３：3　４：4　５：5　６：6　７：7
　　　　　　　８：8　９：9　１０：10　Ｊ：11　Ｑ：12　Ｋ：13

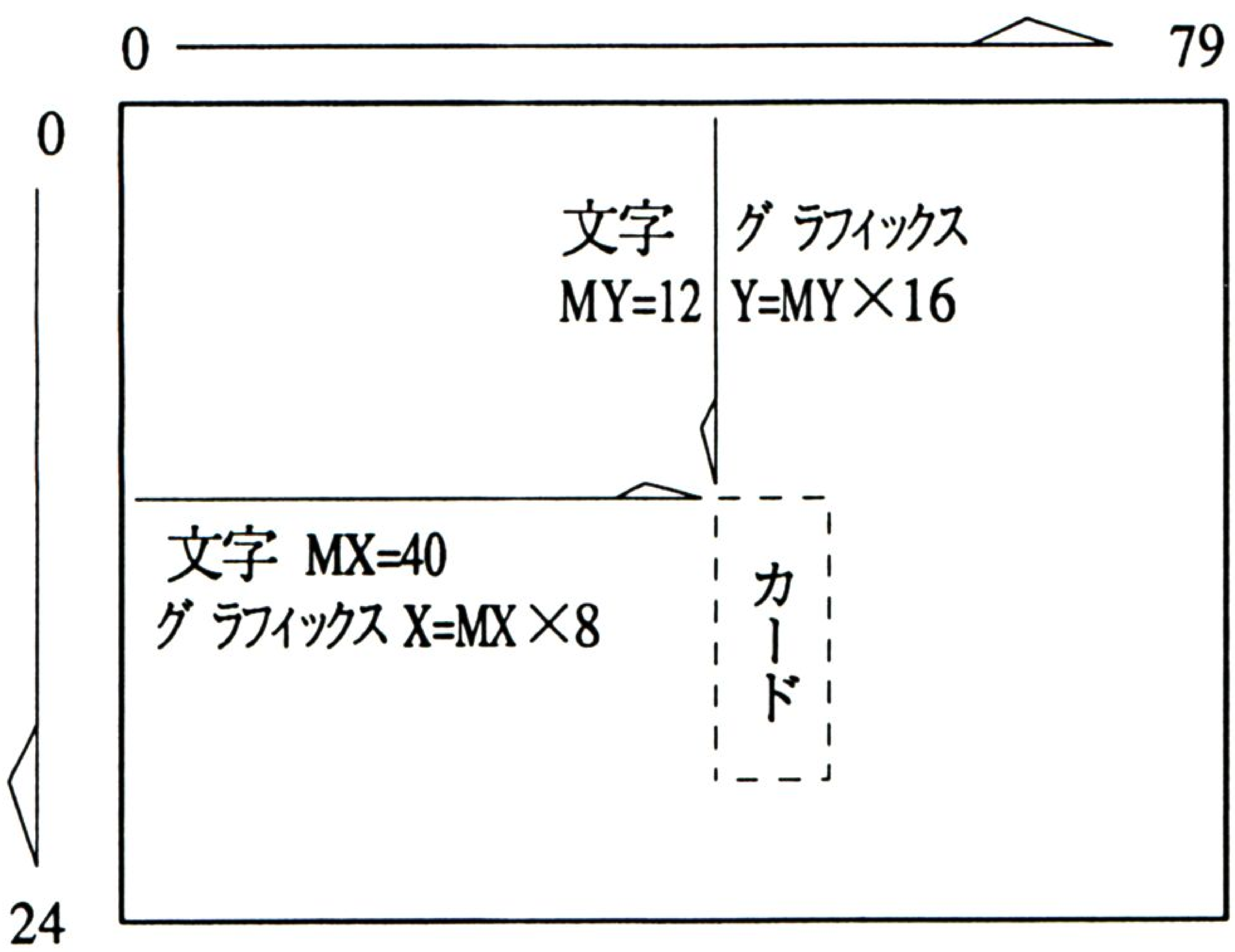
カードの数の文字は、全角文字を使うので、２バイトコードになります。そこで、２バイトの文字列が１３個必要になります。プログラムは次のとおりです。

```
DIM MOJI(1 TO 13) AS STRING * 2
```

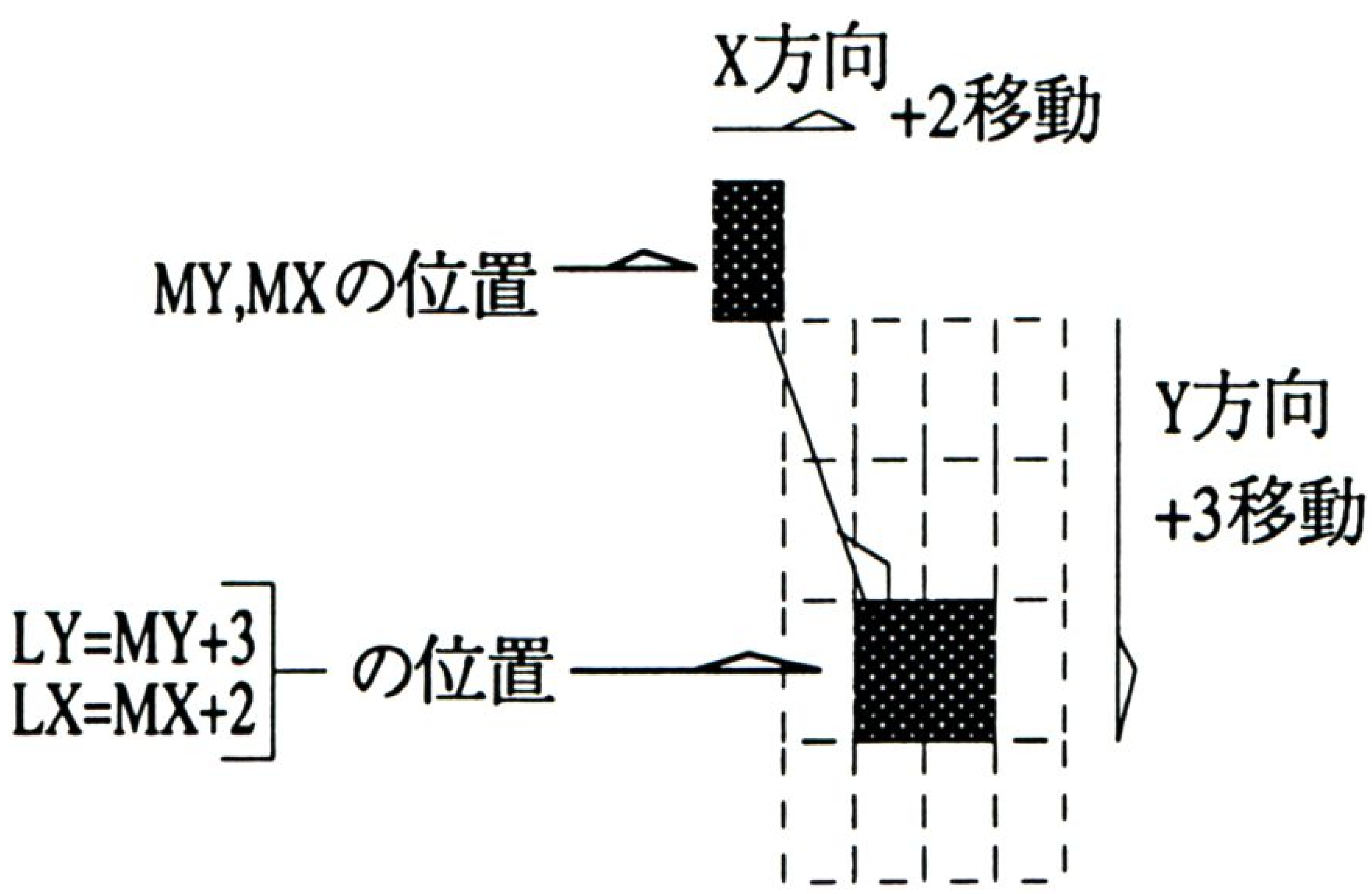
サブプロシージャ内のSHARED・・・・は、メインプロシージャの容器を共同利用するためのステートメントです。本来、サブプロシージャはメインから独立しているのですが、仕事の内容によっては共同利用の方が便利な場合があります。

サブプロシージャCARDHYOUZIは、カードの表示を容易にするためのプログラムです。MX、MYが表示位置ですが、ピクセル数の座標位置ではなく文字を表示するための座標位置を使って、左上の位置を指し示すようにしてあります。カードの文字を表示するには、LX=MX+2、LY=MY+3と位置を調整し、カードの型のキャラクタを表示するには、X=MX*8、Y=MY*16と座標を調整するようになっています。

文字のための座標位置

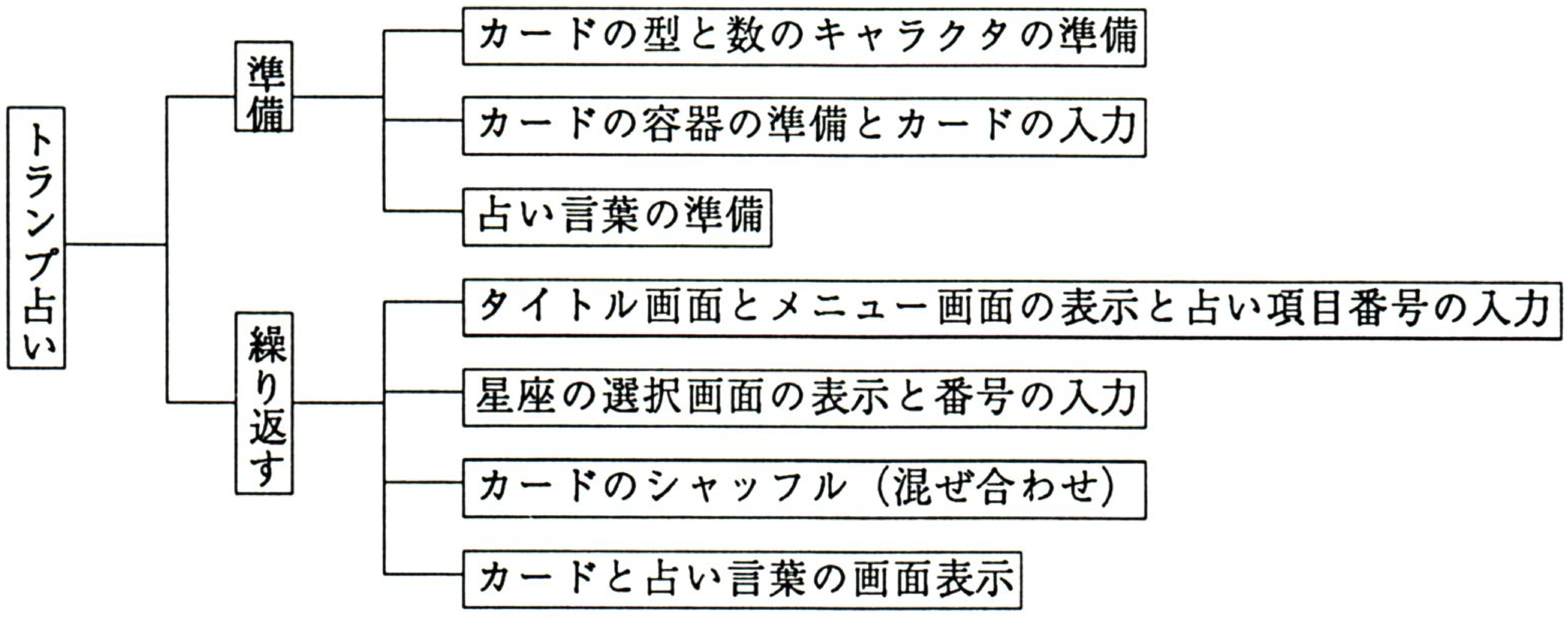


カードの文字の位置関係



4．3　トランプ占いの完成

カードの準備ができたので、いよいよ、トランプ占いを完成させましょう。
トランプ占いのプログラムの構造は次のとおりです。




```
DECLARE SUB CARDHYOUZI (CARD AS ANY, MX AS INTEGER, MY AS INTEGER)
DECLARE SUB CARDYOMI (CARDGR() AS INTEGER, A AS INTEGER, B AS INTEGER)
'占い3      ****トランプ占い　完成****
DEFINT A-B, I-K, T, X-Y
CLS                                          '画面消去
TATE = 64: YOKO = 32                         'トランプの大きさ
B = (4 + 4 * ((YOKO + 7) ¥ 8) * TATE) ¥ 2    'トランプの絵を入れる容器の
                                             '大きさを計算します
DIM DIAMOND(1 TO B) AS INTEGER               'ダイヤの容器を作る
DIM HEART(1 TO B)   AS INTEGER               'ハートの容器を作る
DIM SPADE(1 TO B)   AS INTEGER               'スペードの容器を作る
DIM CLUB(1 TO B)    AS INTEGER               'クラブの容器を作る
DIM URA(1 TO B)     AS INTEGER               'カードの裏面の容器を作る
DIM MOJI(1 TO 13)   AS STRING * 2            'カードの文字の容器を作る

TYPE CARDTYPE                                'カード型のデータを決める
     KATA AS INTEGER                            '型は整数
     SUU  AS INTEGER                            '数は整数
END TYPE                                     '印をつける
DIM CARD(1 TO 52) AS CARDTYPE                'カード５２枚分の容器を作る
DIM CARDW AS CARDTYPE                        'カードを交換するための容器

OPEN "B:CARD.DAT" FOR RANDOM AS #1 LEN = LEN(DIAMOND)    'ファイルを開く
CARDYOMI DIAMOND(), 1, B                     'カードの絵のデータを読み取る
CARDYOMI HEART(), B + 1, B * 2
CARDYOMI SPADE(), B * 2 + 1, B * 3
CARDYOMI CLUB(), B * 3 + 1, B * 4
CARDYOMI URA(), B * 4 + 1, B * 5
CLOSE #1                                     'ファイルを閉じる
```

```
FOR I = 1 TO 13
    READ MOJI(I)                              'カード用の文字を読み取る
NEXT I
                                          'カードの箱の中にカードを入れる
K = 0
FOR I = 1 TO 4                                'カードは４種類ある
    FOR j = 1 TO 13                           '１種類のカードは１３枚ある
        K = K + 1                             '通して何番目のカードか数える
        CARD(K).KATA = I                      '型に１から４の番号を入れる
        CARD(K).SUU = j                       '数に１から１３の番号を入れる
    NEXT j
NEXT I

DIM KOTOBA$(1 TO 5, 0 TO 4)                   '占いの言葉の容器を作る

FOR I = 1 TO 5                                '５種類の項目
    FOR j = 0 TO 4                            '各項目ごとに４つの言葉がある
        READ KOTOBA$(I, j)                    '言葉を読みます
    NEXT j
NEXT I
DO
    CLS                                                  '画面消去
    COLOR 3                                              '水色にする
    PRINT "     ***********************************************************"
    PRINT "     ***********************************************************"
    COLOR 5                                              '紫色にする
    PRINT "        **********        トランプ占い          *********"
    PRINT "        **********   あなたの運勢を占います     *********"
    COLOR 3                                              '水色にする
    PRINT "     ***********************************************************"
    PRINT "     ***********************************************************"
    PRINT
    PRINT
    COLOR 6                                              '黄色にする
    PRINT "                    1: 願いのこと"
    PRINT "                    2: 幸せのこと"
    PRINT "                    3: 就職、進学のこと"
    PRINT "                    4: お金のこと"
    PRINT "                    5: 恋のこと"
    PRINT "                    6: 終了"
    PRINT
    COLOR 4                                              '赤色にする
    PRINT "          どの占いをしますか。番号を入力してください。"
    PRINT
    COLOR 7                                              '白色にする
    DO
        U$ = INKEY$                                      '番号を入力する
    LOOP UNTIL U$ >= "0" AND U$ <= "6"        '範囲内の数字か判定します
```

```
    IF U$ = "6" THEN                              '6ならば終了します
        CLS
        END
    END IF
    PRINT
    PRINT
      "      1:天秤座  2:蠍座  3:射手座  4:山羊座  5:水瓶座  6:魚座"
    PRINT
      "      7:牡羊座  8:牡牛  9:双子座  10：蟹座  11:獅子座  12:乙女座"
    PRINT
    INPUT "          あなたの星座の番号を押してリターンしてください。"
                                                    ; SEIZA
                                            'シャッフル
    RANDOMIZE TIMER
    FOR I = 1 TO 52
        j = 51 * RND + 1
        CARDW = CARD(I)
        CARD(I) = CARD(j)
        CARD(j) = CARDW
    NEXT I

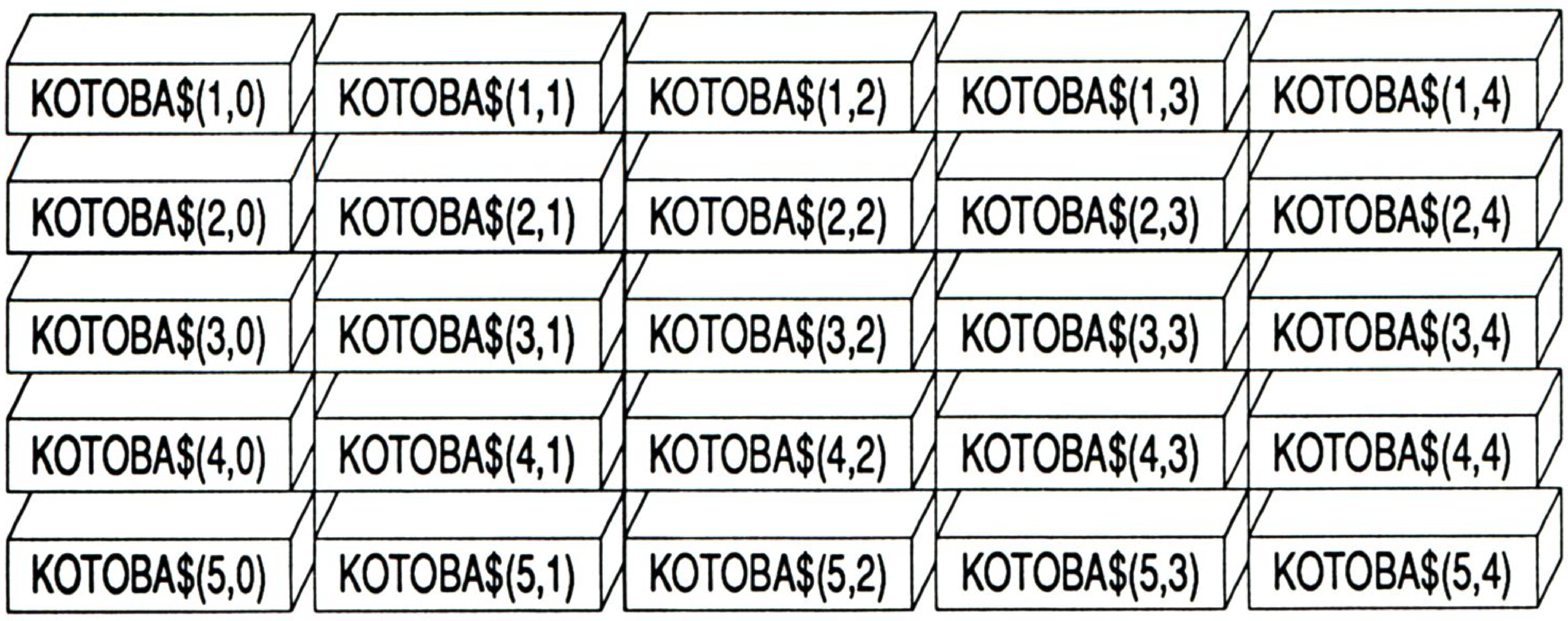
    CLS                                           '画面消去
    CARDHYOUZI CARD(SEIZA), 15, 4                 'カードを15列4行目に表示する

    COLOR 2                                       '緑色にする
    LOCATE 10, 5                                  '10行5列目にカーソル移動
    PRINT "*****  "; KOTOBA$(VAL(U$), 0); "  *****"       '項目を書く
    COLOR 3                                       '水色にする
    LOCATE 13, 5                                  '13行5列目にカーソル移動
    PRINT KOTOBA$(VAL(U$), CARD(SEIZA).KATA)              '占いの言葉を書く

    PRINT
    PRINT
    COLOR 7                                       '白色にする
    PRINT
    PRINT
    PRINT "何かキーを押してください"
    DO
        LOOP UNTIL INKEY$ <> ""                   '何かキーを押せば次に進む
LOOP
END
DATA A , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 ,10, J , Q , K          'カードの数の文字
DATA 願いのこと
DATA あなたの願いを邪魔する人がいます。短気を起こさないこと。
DATA 思うようにはいきませんが、周囲の人たちに助けられます。
DATA すべてのことがかなえられます。
DATA あなたの努力しだいで希望がかなえられます。
DATA 幸せのこと
DATA 幸福は、ながく続きません。忍耐が必要です。
```

```
DATA 自分勝手な行動は、不幸のもとです。自重が大切です。
DATA 何事にも順調に発展し、恵まれます。
DATA 無理をしないで慎重に行動すれば、順調です。
DATA 就職、進学のこと
DATA 面接でのあなたの態度が悪いので合格は難しいでしょう。
DATA 第一希望には合格できませんが、第２第３希望には合格できます。
DATA あなたの実力で、希望のとおり合格できます。
DATA あなたの努力しだいで希望の会社や学校に合格します。
DATA お金のこと
DATA あなたの努力で多額の金が入ります。
DATA 他人に頼ると金を失います。
DATA 金がドンドン入ってきます。
DATA 相当、慎重に行動しないと、金を使い果たしてしまいます。
DATA 恋のこと
DATA 人にねたみをかって邪魔されます。
DATA どこか性格が合わないようです。
DATA 結婚できるでしょう。
DATA 友達関係で終ります。
DEFSNG A-B, I-K, T, X-Y
SUB CARDHYOUZI (CARD AS CARDTYPE, MX AS INTEGER, MY AS INTEGER)
                                                    'カードの表示
                    （省略　前節に同じ）
END SUB
DEFSNG L, X-Y
SUB CARDYOMI (CARDGR() AS INTEGER, A AS INTEGER, B AS INTEGER)
                    （省略　前節に同じ）
END SUB
```

占いの言葉を入れている容器はDIM KOTOBA$(1 TO 5, 0 TO 4)という指定ですが、これは、次の図のような容器です。



これらの容器のうち、KOTOBA$(1, 0)、KOTOBA$(2, 0)･･･KOTOBA$(5, 0)には、占いの項目の名称を入れます。

KOTOBA$(1, 0)に入れた項目についての占いの言葉は、KOTOBA$(1, 1)、KOTOBA$(1, 2)････KOTOBA$(1, 4) に入れてあり、４つの言葉の選択は、選ばれたカードの型番号によって画面へ表示するようにしてあります。

タイトル画面の文字の色を変えるために、COLORステートメントを使っています。

COLOR　色番号　　　のように使い、その色番号は、次のとおりです。

黒の背景色に次の色で文字が表示されます→

黒：0　青：1　緑：2　水：3　赤：4　紫：5　黄：6　白：7

黒の背景色に次の色の帯に黒の文字が表示されます→

黒：8　青：9　緑：10　水：11　赤：12　紫：13　黄：14　白：15

```
DO
    U$ = INKEY$
LOOP UNTIL U$ >= "0"  AND U$ <= "6"
```

入力された１文字を検査して、1から6の範囲以外の文字は受けないようにしています。

```
FOR I  = 1  TO 52
    SWAP CARD(I), CARD(51 * RND + 1)
NEXT I
```

カードをシャッフル（混ぜ合わせる）するプログラムです。52枚のカードが必ず１回以上交換されます。SWAPは、データ内容を交換するステートメントです。

51*RND+1は、一様乱数を使って1から52までの数を計算します。これによって、I番目と51*RND+1番目のカードを交換することになるのです。

```
CARDHYOUZI CARD(SEIZA), 15  ,4
```

シャッフルしたカードの星座番号番目のカードを、画面の15行4列目の位置に表示します。

```
PRINT "*****   ";  KOTOBA$(VAL(U$), 0);  "   *****"
```

選択された占いの項目の名称を表示します。U$は１文字入力で入力された文字型のデータなので、VALで文字を数値に変えて使っています。

```
PRINT KOTOBA$(VAL(U$), CARD(SEIZA).KATA)
```

選択された項目と、入力された星座番号番目のカードの型に対応した占いの言葉を表示します。

４．４　占い専用のフロッピーディスク

プログラムを実際に利用する際に、その都度Ｑｕｉｃｋ　ＢＡＳＩＣを必要する方法では、使いにくいので、占い専用のフロッピーディスクを作ることにします。

占い専用のフロッピーディスクを作る手順の概要は次のとおりです。

1. 新しいフロッピーディスクを用意して、システム付きでフォーマットをします。

Ａドライブへ ＭＳ－ＤＯＳのシステムディスクをセットし、Ｂドライブへ新しいフロッピーディスクをセットして、ＭＳ－ＤＯＳを起動の後に、次のように入力します。

```
A>FORMAT B: /S
```

2. プログラムで作成したカードのキャラクタのファイル（CARD.DAT)を新しいフロッピーディスクへコピーします。

Ａドライブへ CARD.DATの入ったフロッピーディスクをセットし、Ｂドライブへ新しいフロッピーディスクをセットして次のように入力します。

```
A>COPY A:CARD.DAT B:
```

3. 占いのプログラムの中のB:CARD.DATをA:CARD.DATと変更します。

詳細は、次の節で説明します。

4. Ｑｕｉｃｋ　ＢＡＳＩＣを使って、占いのプログラムから実行形式のプログラムを作成し、新しいフロッピーディスクへコピーします。

詳細は、次の節で説明します。

5. 新しいフロッピーディスクへCONFIG.SYS及びAUTOEXEC.BATのファイルをつくります。

詳細は、次節以降で説明します。

このようにして作成した占い専用のフロッピーディスクは、パソコンの電源を入れてＡドライブへセットするだけで、直接実行が開始されて、占いのタイトル画面が表示されるようになります。

4．5　実行用のプログラムの作成

まず、占いのプログラムの一部を改造します。

これまでAドライブへQｕｉｃｋ　ＢＡＳＩＣのシステムをセットして使用していたので、カードのキャラクタのファイルをBドライブへ置いて読み取るようにしてきました。しかし、占い専用のフロッピーディスクはAドライブへセットして使用することにしますから、カードのキャラクタのファイルもAドライブから読み取ることができるように、B:をA:に修正する必要があるのです。

改造するステートメントは次のとおりです。

改造前

```
OPEN "B:CARD.DAT" FOR RANDOM AS #1 LEN =        LEN(DIAMOND)
```

改造後

```
OPEN "A:CARD.DAT" FOR RANDOM AS #1 LEN =        LEN(DIAMOND)
```

次に、いよいよ実行形式のプログラムの作成を行います。

（占いのプログラムは、B:URANAI.BASという名称で保存してあるものとします。）

1. マウスカーソルを［R/実行］に合わせて左クリックすると、次のウインドウが表示されます。

```
S/スタート                SHIFT+F5
R/リスタート
N/継続                          F5
C/COMMANDの編集 ...

X/実行ファイル作成 ...
L/ライブラリ作成 ...

M/メインモジュールの設定 ...
```

2. マウスカーソルを［X/実行ファイル作成 ...］に合わせて左クリックすると、次のウインドウが表示されます。

> **X/実行ファイル作成**
>
> 実行ファイル名とコンパイルスイッチを設定してください
>
> N/実行ファイル名: B:URANAI.EXE
>
> [ ] D/デバッグ　　　(*) X/ランタイム分離型
>
> 　　　　　　　　　　( ) A/独立型
>
> < M/実行ファイル作成 >　< E/実行ファイル作成後終了 >
>
> < 取消 >　< H/ヘルプ >

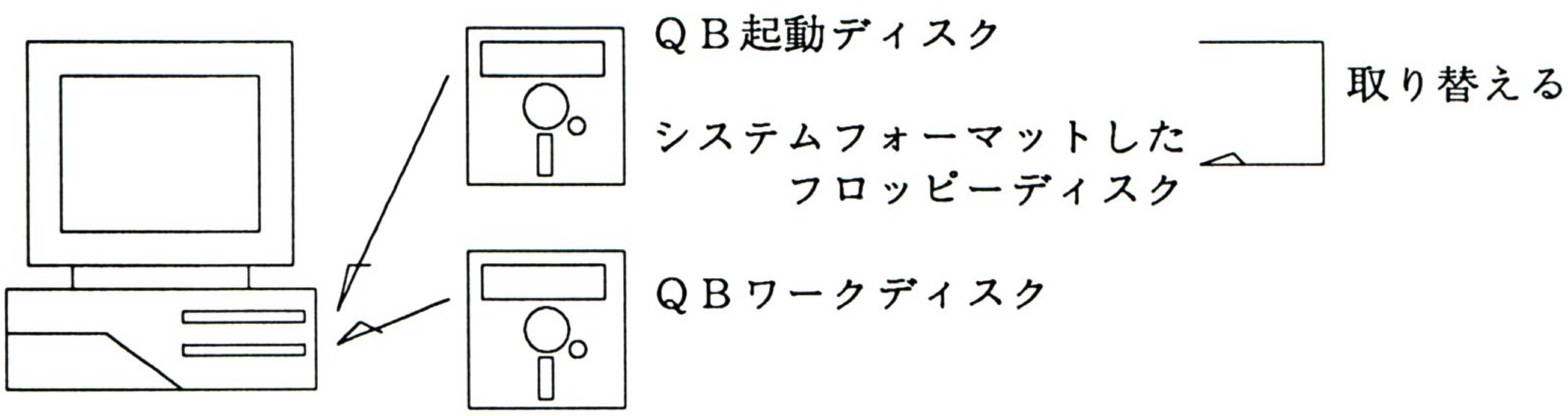
3. 実行ファイル名を入力する欄にマウスカーソルを合わせて、キーボードからB:URANAI.EXEと入力します。

4. ［( ) A/独立型］の( )にマウスカーソルを合わせて、左クリックします。
すると、(*)と表示されます。

5. < M/実行ファイル作成 >にマウスカーソルを合わせて、左クリックします。
作業が開始し、Ｂドライブのフロッピーディスクに、URANAI.EXEという実行形式のプログラムが作られます。

6. Ｑｕｉｃｋ　ＢＡＳＩＣを終了してから、４．４でシステムフォーマットしたフロッピーディスクをＡドライブにセットし、次のように入力してURANAI.EXEをコピーします。

```
A>COPY B:URANAI.EXE A:
```

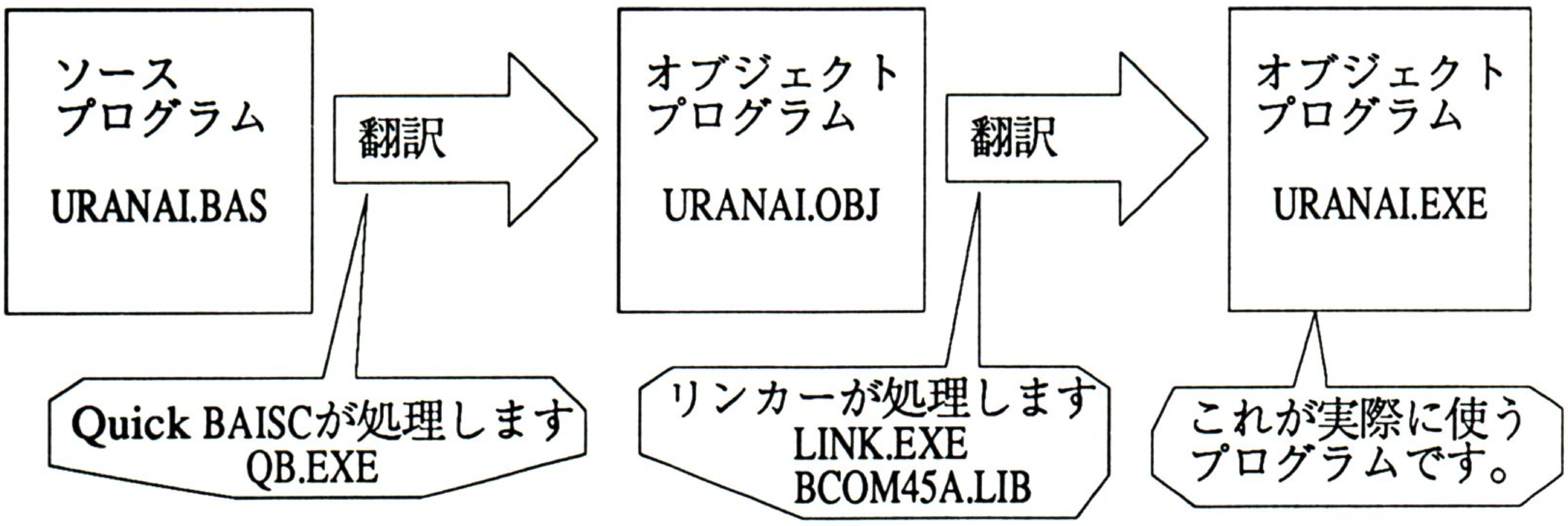


今行った作業は、私達が作ったプログラムを、パソコンが直接的に使えるように翻訳したプログラムを作成したのです。このプログラムのことを、一般にオブジェクトプログラム、実行用のプログラムなどといいます。それに対して、最初に作ったプログラムをソースプログラムといいます。

これらのプログラムは、プログラム名の３桁の拡張子で区別できるようにしています。

ソースプログラム	URANAI.BAS
オブジェクトプログラム（相対形式、リンケージの前）	URANAI.OBJ
オブジェクトプログラム（実行形式、リンケージの後）	URANAI.EXE

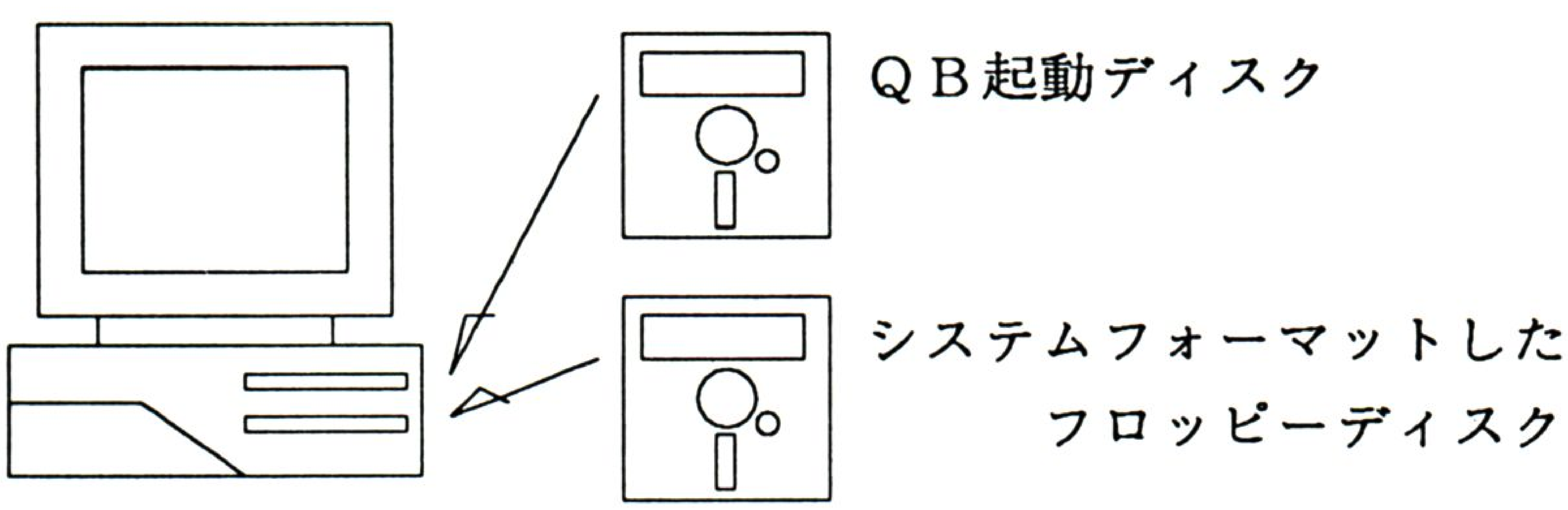
実は、ソースプログラムから直接に実行用のプログラムを作るのではなく、もう一つの工程があります。次の図を参考にしてください。



蛇足ながら、もう一つ付け加えます。もしも**4.** で［( ) X/ランタイム分離型］を選択して実行用のプログラムを作った場合は、URANAI.EXEのプログラムを小さくできるメリットがあります。ただし、この場合はBRUN45A.LIBの助けを借りて動くので、URANAI.EXE及びBRUN45A.EXEをシステムフォーマットしたフロッピーディスクにコピーして使う必要があります。

BRUN45A.EXEのコピーは、Ａドライブへ ＱＢ起動ディスクを、Ｂドライブへシステムフォーマットしたフロッピーディスクをセットして次のように入力します。

```
A>CD BIN
A>COPY BRUN45A.EXE B:
```



4．6　自動的に占いが始まる

さて、新しいフロッピーディスクには、占いのプログラムを動かすためのファイルがコピーされてきました。まず、その内容を確認します。

確認の方法は、Ａドライブに４．５章で作成したフロッピーディスクをセットして次のように入力して↵を押します。

```
A>DIR
```

DIRは、フロッピーディスクのファイルのリストを見る機能をもつ命令です。

すると、新しいフロッピーディスクにあるファイルのリストが表示されます。

表示されるファイルは、次の３つのファイルです。

COMMAN.COM
CARD.DAT
URANAI.EXE

（ただし、URANAI.EXEがランタイム分離型の場合は、BRUN45A.EXEが必要である。）

次に、自動的に占いのプログラムが起動するための仕掛けを加えます。

次のように入力します。１行目を入力したら↵で改行し、続いて２行目を入力したらCTRLとＺを同時に押してさらに↵を押します。

```
A>COPY CON AUTOEXEC.BAT
URANAI
```

これで完成です。ＲＥＳＥＴを押してください。自動的に占いのタイトル画面が表示され、ここから占いが始まります。

6を押して、終了させてから、改めて、DIR↵を押してください。

今度は、AUTOEXEC.BATというファイルが増えています。このファイルは、自動的にプログラムを起動する仕掛けなのです。

COPY CON....のCONは、コンソールのCONで、ここでは、キーボードから入力した内容を、AUTOEXEC.BAT という名称のファイルにコピーしなさいという意味です。

第❺章

電話帳

シーケンシャルファイル・ランダムアクセスファイルの基本を学びます。

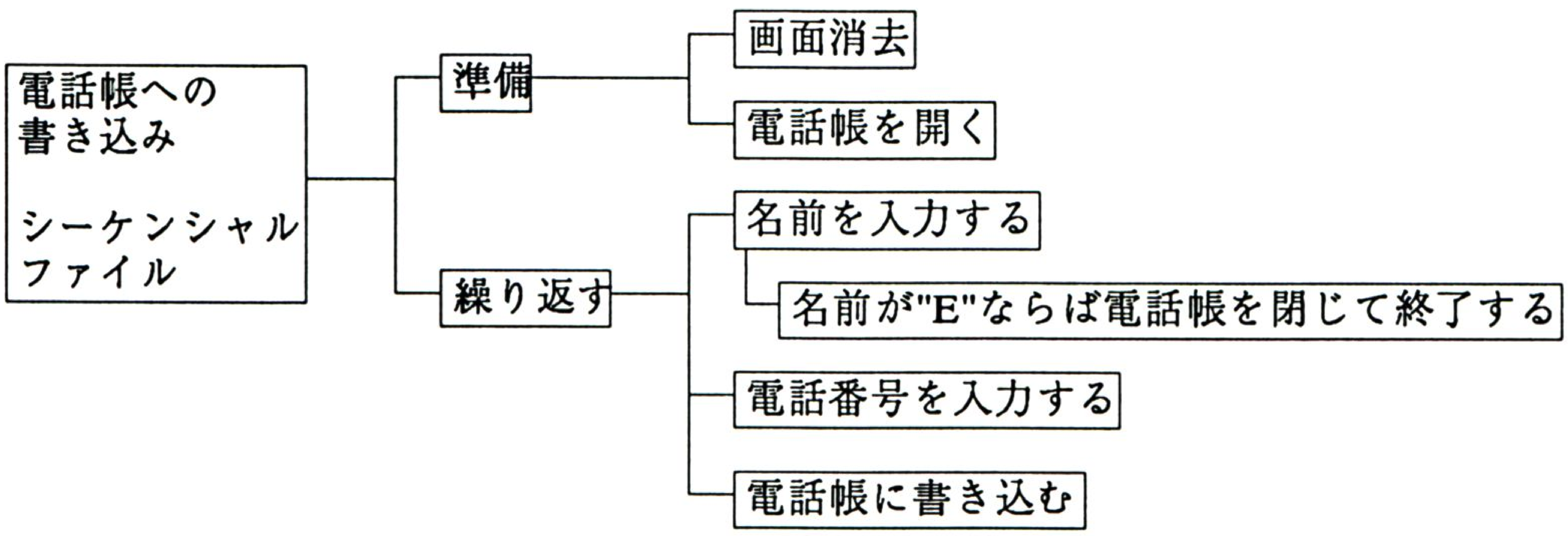
５．１　電話帳の読み書き

電話帳は、誰もが日常に使う小道具でデータの内容について特に説明する必要もないので、パソコンのプログラムの学習のために格好な教材です。個人の電話帳自体は小さい手帳に書いておけば十分間に合うのですが、パソコンの電話帳を拡大して、得意先管理システムに発展させることも可能だと思います。

電話帳ファイルの作り方として、まずシーケンシャルファイルとしてつくる方法があります。

この方法は、他のいろいろなソフトで電話帳のデータを利用できる便利な形式です。しかし、ＶＴＲのテープと同様にテープの中の一部のみを容易に書き換えることができないということ、読み書きが低速になること、データの量が大きくなること、ある一部を読むためにも最初から順序に読む必要があることなど、多くの欠点があります。

シーケンシャルファイルへの書き込みのプログラムの構造は次のとおりです。




```
'電話帳１　　シーケンシャルファイルへの書き込み
CLS                                          '画面消去
OPEN "B:DENWA.DAT" FOR OUTPUT AS #1          '電話帳を開く
DO                                           '繰り返す
    INPUT "名前(終了:E)：", NAMAE$           '名前の入力(終了時はE)
    IF NAMAE$ = "E" THEN                     '名前がEならば
        CLOSE #1                             '電話帳を閉じて
        END                                  '終了
    END IF
    INPUT "電話番号：", BANGOU$              '電話番号の入力
    WRITE #1, NAMAE$, BANGOU$                '電話帳への書き込み
LOOP
```

このプログラムで次のデータを入力すると、フロッピーディスクには次のように記憶されています。

1．データの例

名　　前	電話番号
佐藤　花子	４５－７８９４
井上　幸太郎	７８－６５２３
山田　健一	１２－４５３３
佐々木　康子	８５－７５４７
近藤　幸枝	５６－８５５２

2．フロッピーディスクへの記憶の状態

```
22 8D B2 93 A1 81 40 89-D4 8E 71 22 2C 22 82 53    "佐藤　花子"," 4
82 54 81 7C 82 56 82 57-82 58 82 53 22 0D 0A 22    5－7894".."
88 E4 8F E3 81 40 8D 4B-91 BE 98 59 22 2C 22 82    井上　幸太郎"," 7
56 82 57 81 7C 82 55 82-54 82 51 82 52 22 0D 0A    8－6523"..
22 8E 52 93 63 81 40 8C-92 88 EA 22 2C 22 82 50    "山田　健一"," 1
82 51 81 7C 82 53 82 54-82 52 82 52 22 0D 0A 22    2－4533".."
8D B2 81 58 96 D8 81 40-8D 4E 8E 71 22 2C 22 82    佐々木　康子"," 8
57 82 54 81 7C 82 56 82-54 82 53 82 56 22 0D 0A    5－7547"..
22 8B DF 93 A1 81 40 8D-4B 8E 7D 22 2C 22 82 54    "近藤　幸枝"," 5
82 55 81 7C 82 57 82 54-82 54 82 51 22 0D 0A 00    6－8552"...
```

左のブロックは１６進数で表記してあり、右のブロックはコードで対応する文字で表記してあります。

データをダブルクォーテーションで区切る形式ですから、他のソフトでも容易に読むことができます。

```
OPEN "B:DENWA.DAT" FOR OUTPUT AS #1
```

電話帳を、シーケンシャルファイルとして新たに作成するステートメントです。

B:は、Ｂドライブのフロッピーディスクやハードディスクに、電話帳をつくることを意味しています。

DENWA.DATはファイル名で、プログラムをつくる人が任意に名前を付けることができますから、分りやすい名前を付けることが大切です。ファイル名のうち、DATの部分は拡張子といいます。拡張子は一般的にファイルの性格を表し、例えばこのDATはファイルの内容がデータであることを表しています。他にもDOC（操作手引な

ど説明書のファイル）、BAS（ＢＡＳＩＣのソースプログラム）などがあり、中でもBAT（バッチファイル）、EXE、COM（実行型のプログラム）、などは必ずこの名前でなければなりません。

FOR OUTPUTは「出力（書く）のための」ということです。しかし、同じ出力でもすでに存在するファイルに追加して書き込む場合には、FOR APPENDとします。

また、入力（読む）の場合は、FOR INPUT とします。

```
INPUT "名前(終了:E)：", NAMAE$
```

キーボードから入力したデータをNAMAE$という文字型の容器に入れるということです。

"名前(終了:E)："は、入力する内容について案内するメッセージです。

名前は一般に漢字で入力しますので、入力するよう求められたら CTRL を押しながら XFER を押して漢字入力モードにしてから、名前や電話番号を入力します。名前の入力の終了時には、ここで半角の大文字のEを入力するよう求めています。大文字と小文字と区別しているので、小文字のeを入力すると名前がeということになってしまいます。終了のためにEを入力する際は、再び CTRL を押しながら XFER を押して、英数モードに戻してから入力します。

"とNAMAE$の間に，（カンマ）がありますが、これを；（セミコロン）に替えると、画面上で入力を求める際に?を表示します。

```
WRITE #1, NAMAE$, BANGOU$
```

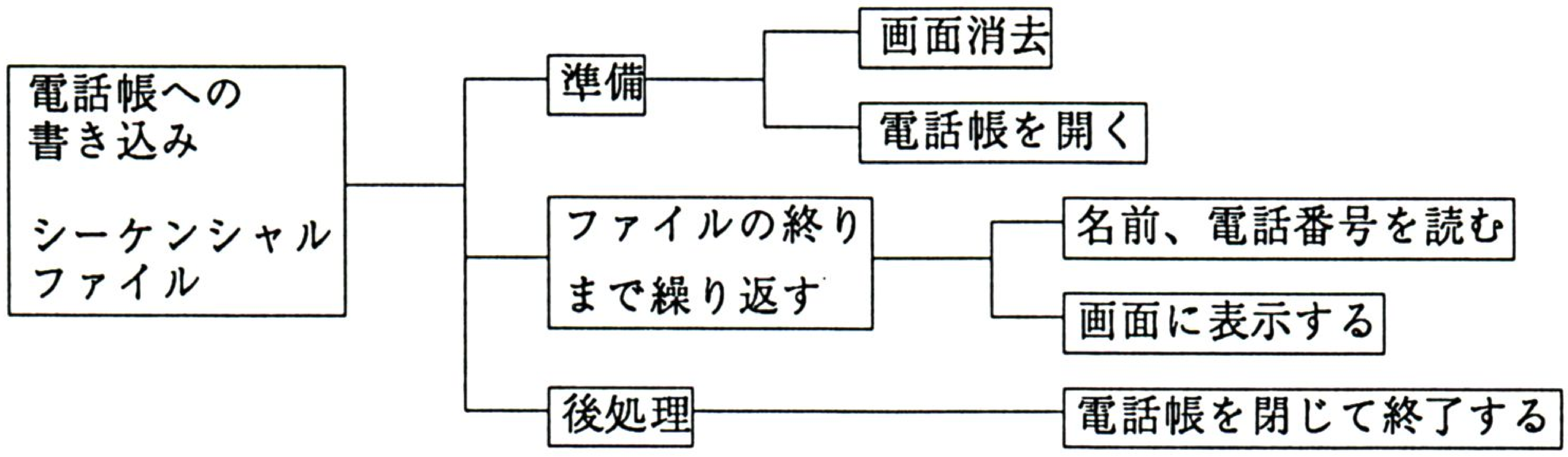
シーケンシャルファイル#1へNAMAE$とBANGOU$の内容を書き込むためのステートメントです。

#1は、ファイルを扱う際の番号で、OPENステートメントで決めたものを使います。従って、OPENステートメントでAS #2とあれば、WRITE #2, ‥‥となるのです。

WRITEステートメントの特徴は、ファイルの中のデータの表記において、文字列は"（ダブルクォーテーション）で囲まれ、データの区切りには　,（カンマ）が入れられるところにあります。こうすれば、他のいろいろなパソコンソフトでも、このファイルを利用できるのです。

もしもPRINT #1,‥‥とすると、画面に表示されるときと同様に区切り記号がないので、データを読み取りにくいファイルになります。

次に、シーケンシャルファイルの電話帳を読んで画面へ表示します。
このプログラムの構造は次のとおりです




```
'電話帳２　　シーケンシャルファイルを読んで画面表示
CLS                                                      '画面消去
OPEN "B:DENWA.DAT" FOR INPUT AS #1                       '電話帳を開く
DO UNTIL EOF(1)                      'ファイルのデータがなくなるまで繰り返す
    INPUT #1, NAMAE$, BANGOU$                            '電話帳を読む
    PRINT NAMAE$, BANGOU$                                '画面表示
LOOP
CLOSE #1                                                 '電話帳を閉じる
END
```

```
DO UNTIL EOF(1)
```

「ファイル#1が終りになるまで繰り返す」ということです。
EOF(1)が「ファイル#1の終り」の信号を出す関数です。EOF(2) ならば「ファイル#2の終り」ということです。
UNTIL EOF(1)は、DO····LOOP UNTIL EOF(1) のように、LOOPの後に書いても同じです。

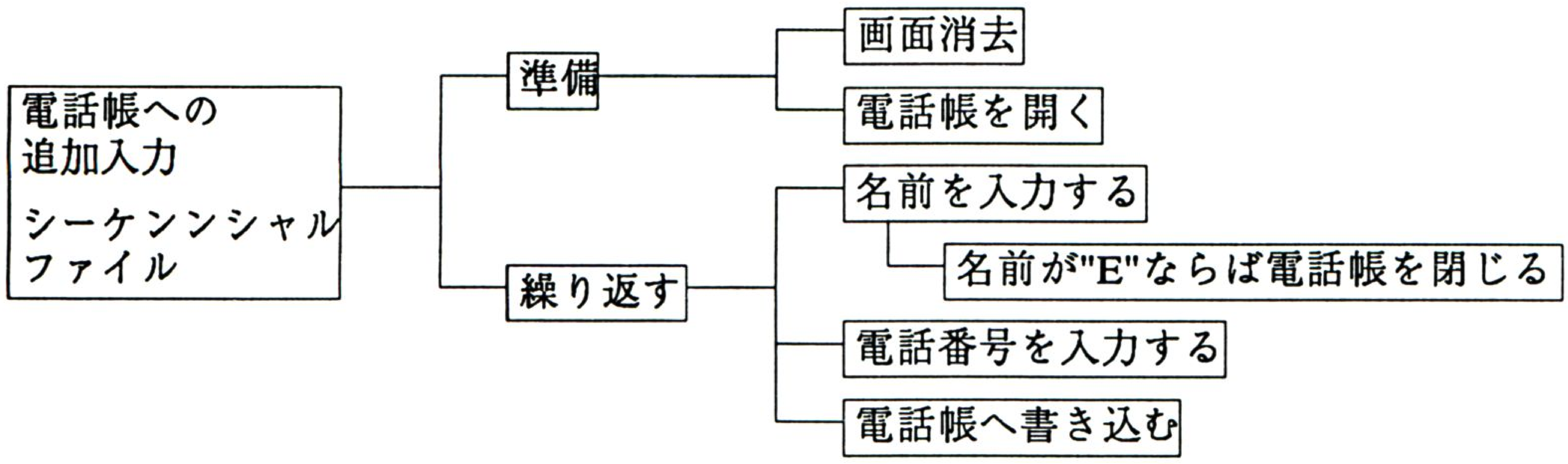
```
INPUT #1, NAMAE$, BANGOU$
```

「名前、電話番号をファイル#1から読む」ことです。
シーケンシャルファイル#1へWRITE #1,···で書き込んだファイルを読むときにはINPUT #1,····を使うのです。

なお、PRINTをLPRINTに変更すると、印刷されます。

次に、シーケンシャルファイルの電話帳にデータを追加入力するプログラムを作ります。

このプログラムの構造は次のとおりです。




```
'電話帳３　　シーケンシャルファイルへの追加入力
CLS                                                    '←画面消去
OPEN "B:DENWA.DAT" FOR APPEND AS #1                    '←電話帳を開く
DO                                                     '←繰り返す
    INPUT "名前(終了:E)：", NAMAE$                     '←名前の入力(終了時はE)
    IF NAMAE$ = "E" THEN                               '← 名前がEならば
        CLOSE #1                                       '←電話帳を閉じて
        END                                            '←終了
    END IF
    INPUT "電話番号：", BANGOU$                        '←電話番号の入力
    WRITE #1, NAMAE$, BANGOU$                          '←電話帳への書き込み
LOOP
```

OPENステートメントのFOR APPENDは、シーケンシャルファイルへの追加入力をする際に使います。

その外は、電話帳１のプログラムに同じです。

以上のように、シーケンシャルファイルの読み書きは単純で分りやすいのですが、データの一部を訂正するためには、別にファイルを作成してこれにコピーしながら訂正部分を探して書き直すという複雑な手順を必要とします。また、多量のデータの中からある特定のデータを読む場合も、最初からひとつひとつ読みながら探す必要があります。

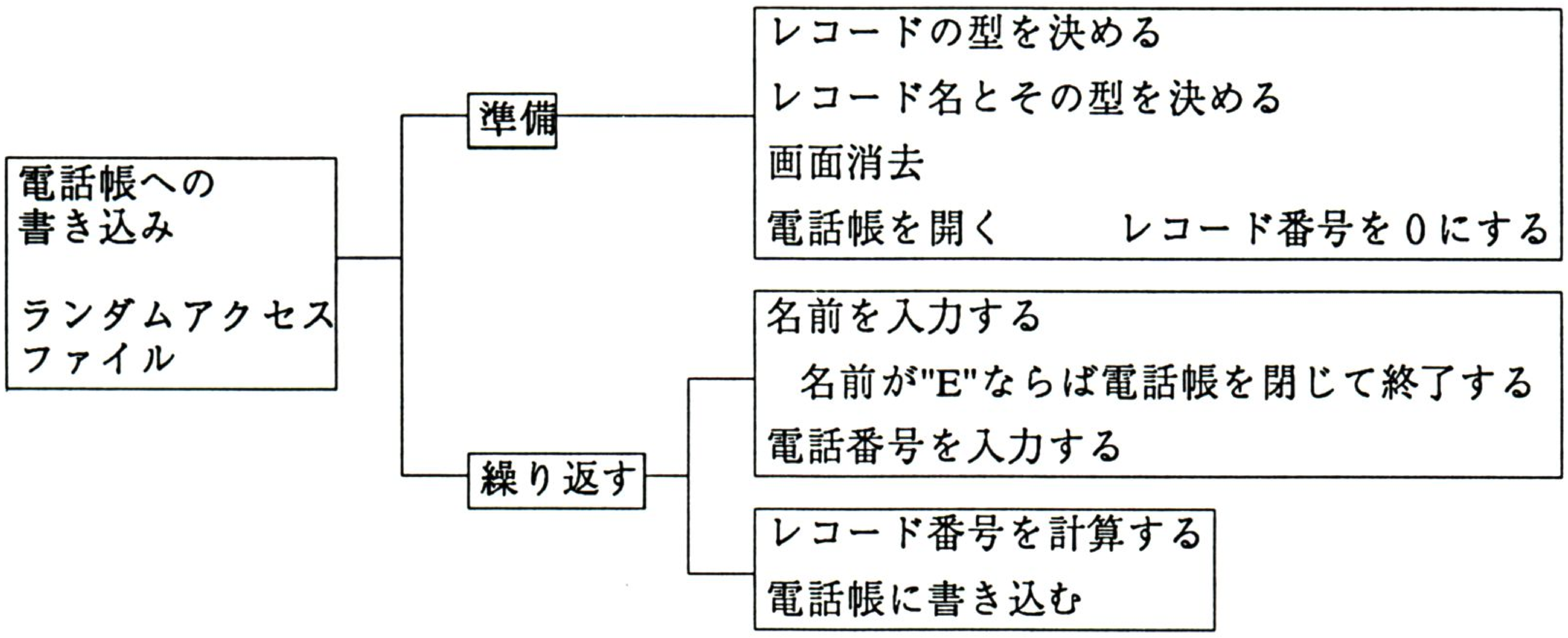
このようなことから、次に学ぶランダムアクセスファイルが、使われるようになりました。

5．2　ランダムアクセスファイル

電話帳をランダムアクセスファイルで作ります。

ランダムアクセスファイルは、名前のとおりランダムにアクセスできる（任意の位置のデータを読み書きできる）ファイルですから、データの訂正や検索に便利です。また、読み書きの速度も比較的速いようです。

ランダムアクセスファイルへの書き込みのプログラムの構造は次のとおりです。




```
'電話帳4　ランダムアクセスファイルへの書き込み
DEFINT I                                            '整数型変数：レコード番号
TYPE DENWATYPE                                      'DENWATYPEというタイプを決める
    NAMAE  AS STRING * 14                           '名前は14桁（全角7字）
    BANGOU AS STRING * 16                           '電話番号は16桁（全角8字）
    S      AS INTEGER                               'レコードの削除のシルシ
END TYPE
DIM DENWA AS DENWATYPE                              'レコードのタイプはDENWATYPE
CLS                                                 '画面消去
OPEN "B:DENWA.DAT" FOR RANDOM AS #1 LEN = LEN(DENWA)       '電話帳を開く
I = 0                                               'レコード番号を0にする
DENWA.S = -1                                        'レコードは有効
DO                                                  '人数分繰り返す
    INPUT "名前(終了:E)：", N$                      '名前の入力(終了時はE)
    IF N$ = "E" THEN                                '　名前がEならば
        CLOSE #1                                    '電話帳を閉じて
        END                                         '終了
    END IF
    DENWA.NAMAE = N$                                'N$を名前へコピー
    INPUT "電話番号：", DENWA.BANGOU                '電話番号の入力
    I = I + 1                                       'レコード番号へ1を加える
    PUT #1, I, DENWA                                '電話帳への書き込み
LOOP
```

```
TYPE DENWATYPE
    ....
    S          AS INTEGER
END TYPE
```

レコード型の型を設定するステートメントです。レコードとは、1件分のデータをいいます。この1件分のデータを構成する各項目の名前と大きさをまとめて、レコードの型として設定しているのです。

DENWATYPEは、プログラムを作る人が任意に決める型の名前です。名前の終りにTYPEを付ける必要はありませんが、これによって他の型の名前と区別しています。

Sは、レコードの削除のシルシです。このレコードが生きているときは-1(マイナス1)を、削除のシルシを0(ゼロ)とします。実は、ある特定のレコードだけを実際に削除するには複雑な手順を必要とするので、ここでは、削除のシルシを付けて削除の扱いにします。これによって、削除したレコードの復活も可能になります。

```
DIM DENMA AS DENWATYPE
```

実際のレコードをDENWAと名付けて、このレコードの構造をDENWATYPEであると設定しています。そこで、DENWAは、NAMAE、BANGOU、Sのデータを包括したレコード名ということになり、これらのデータのすべてをDENWAという名前で取り扱うことができます。また、NAMAEのみを使う場合はDENWA.NAMAEという名で、BANGOUのみを使う場合はDENWA.BANGOUという名で扱うことができます。

```
OPEN "B:DENWA.DAT" FOR RANDOM AS #1  LEN = LEN(DENWA)
```

ランダムアクセスファイルの使用を開始するステートメントです。

ランダムアクセスファイルには、レコードやデータの各項目間に区切り記号はなく、最初に定めたレコードの大きさをもとにして所定のレコード位置に読み書きするようになっています。そこで、LEN＝でレコードの長さを書きます。右辺のLENは、レコードの長さを計算する関数です。従ってLEN＝LEN(DENWA)は、「レコードの長さはDENWAの長さです」ということです。

```
I = I + 1
```

ランダムアクセスファイルは、データを端から順に入出力しないので、入出力の際にレコードの位置を指示する必要があります。これをここではレコード番号と呼んでいます。レコード番号の容器は、最初はゼロにしておき（I＝0）、データを入

力するたびに１を加える（I ＝ I ＋ 1）ようにしました。

```
PUT #1, I, DENWA
```

ランダムアクセスファイル#1に書き込むステートメントです。Iは書き込む位置で、DENWAは書き込むレコードです。

ランダムアクセスファイルの記憶の状態

レコード名	DENWA	DENWA	DENWA	DENWA	DENWA	DENWA	DENWA	DENWA	DENWA	DENWA
レコード番号	1	2	3	4	5	6	7	8	9	10

DENWA.NAMAE	DENWA.BANGOU	DENWA.S
文字列　14	文字列　16	整数　2

１レコードの長さは32バイトです
14+16+2

```
 佐     藤    全角空白  花     子       半角空白        4
8D B2 93 A1 81 40 89 D4-8E 71 20 20 20 20 82 53   佐藤　花子　　4
82 54 81 7C 82 56 82 57-82 58 82 53 20 20 FF FF   5－7894　..
 5     －     7     8     9     4    半角空白 整数-1
88 E4 8F E3 81 40 8D 4B-91 BE 98 59 20 20 82 56   井上　幸太郎　7
82 57 81 7C 82 55 82 54-82 51 82 52 20 20 FF FF   8－6523　..
8E 52 93 63 81 40 8C 92-88 EA 20 20 20 20 82 50   山田　健一　　1
82 51 81 7C 82 53 82 54-82 52 82 52 20 20 FF FF   2－4533　..
8D B2 81 58 96 D8 81 40-8D 4E 8E 71 20 20 82 57   佐々木　康子　8
82 54 81 7C 82 56 82 54-82 53 82 56 20 20 FF FF   5－7547　..
8B DF 93 A1 81 40 8D 4B-8E 7D 20 20 20 20 82 54   近藤　幸枝　　5
82 55 81 7C 82 57 82 54-82 54 82 51 20 20 FF FF   6－8552　..
```

漢字は、それぞれの漢字コード（シフトＪＩＳコード）で記憶しています。

整数の-1は、フロッピーディスク上では、-1の文字コードではなく-1の値をFF FFと表現して記憶しています。

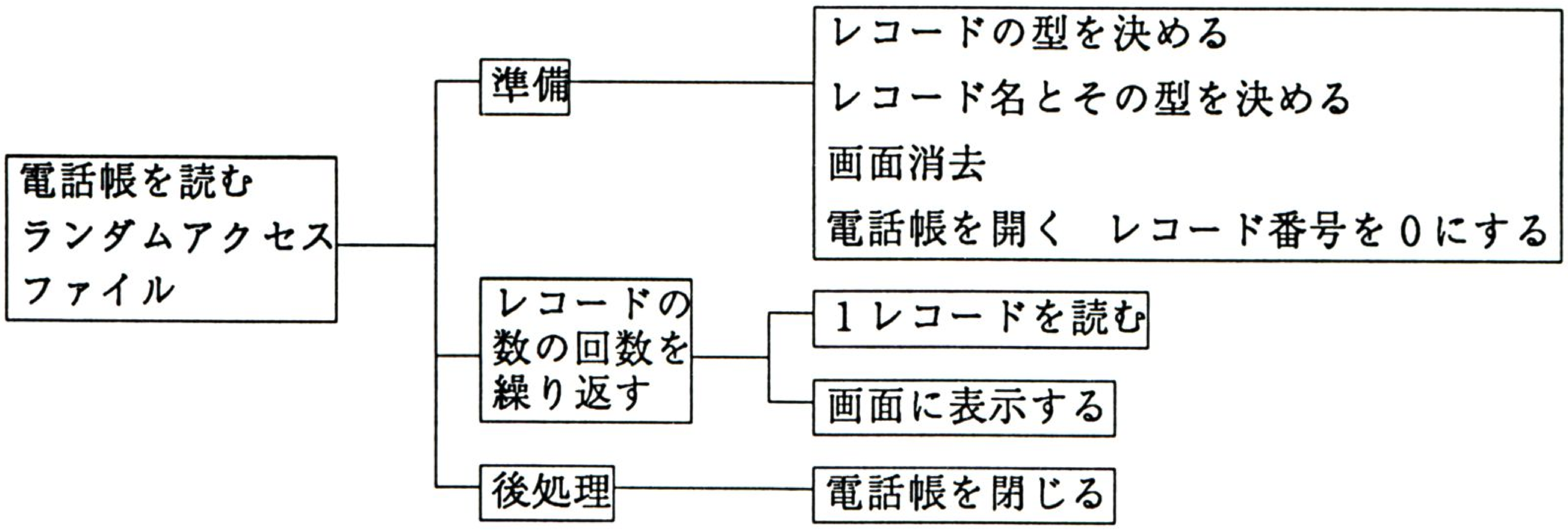
では、他の整数はどのように記憶されるのでしょうか。

5→05 00　4→04 00　3→03 00　2→02 00　1→01 00　0→00 00

-1→FF FF　-2→FE FF　-3→FD FF　-4→FC FF　-5→FB FF

5は、00 05を左右入れ換えて05 00と記憶します。マイナスの整数は補数で表します。10進数では1の補数は9ですが、16進数ではFなので-1はFF FFになります。-2の補数はFF FEなのですが、パソコンの都合で左右入れ換えてFE FFというように記憶しています。

次に、ランダムアクセスファイルを読んでその内容を画面に表示します。このプログラムの構造は次のとおりです。




```
'電話帳5　　ランダムアクセスファイルを読んで画面表示
CLS                                                     '画面消去
DEFINT I                                                'レコード番号
TYPE DENWATYPE                                          'DENWATYPEというタイプを決める
    NAMAE  AS STRING * 14                               '名前は14桁（全角7字）
    BANGOU AS STRING * 16                               '電話番号は16桁（全角8字）
    S      AS INTEGER                                   'レコードの削除のシルシ
END TYPE
DIM DENWA AS DENWATYPE                                  'レコードのタイプはDENWATYPE

OPEN "B:DENWA.DAT" FOR RANDOM AS #1 LEN = LEN(DENWA)    '電話帳を開く
FOR I = 1 TO LOF(1) ¥ LEN(DENWA)                        'レコードの数だけ繰り返す
    GET #1, I, DENWA                                    '読む
    PRINT I; DENWA.NAMAE; DENWA.BANGOU; DENWA.S         '画面へ表示
NEXT I
CLOSE #1
END
```

```
OPEN "B:DENWA.DAT" FOR RANDOM AS #1 LEN = LEN(DENWA)
```

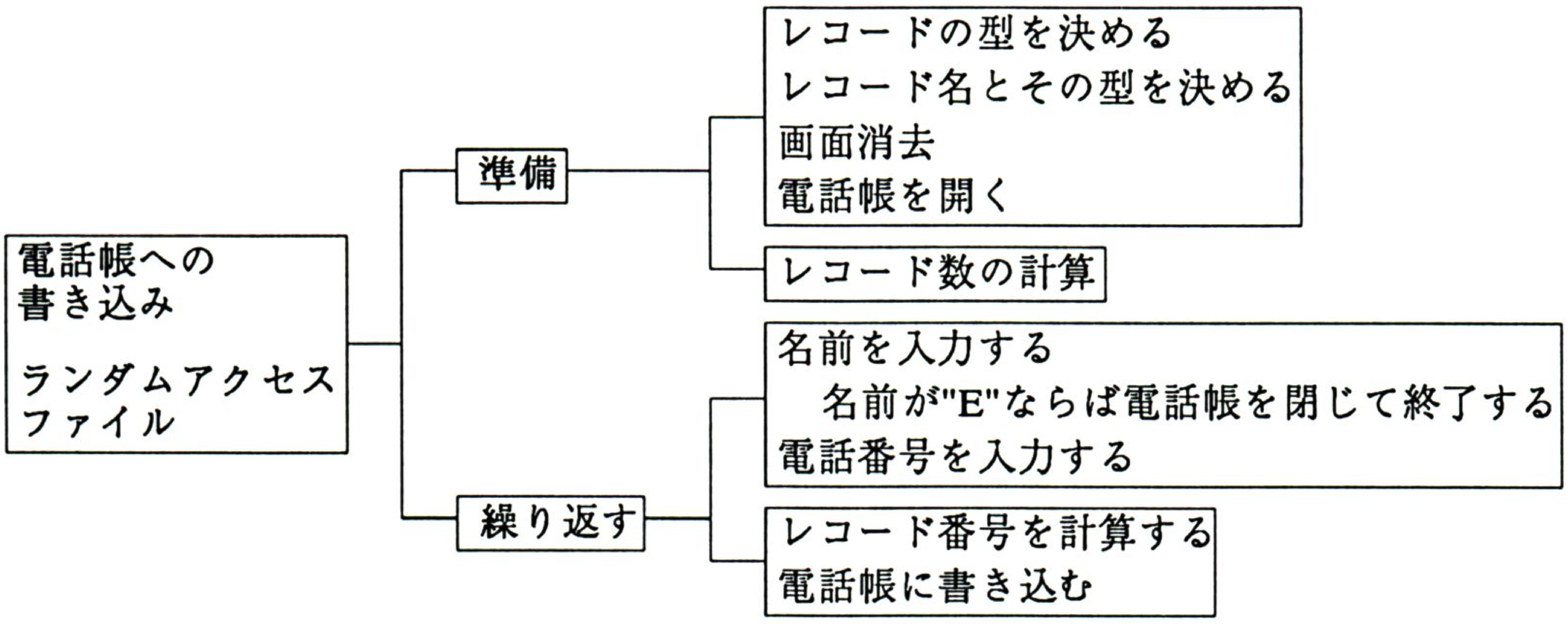
ランダムアクセスファイルの場合は、読み込みも書き込みも開き方は同じです。

```
FOR I = 1 TO LOF(1) ¥ LEN(DENWA) · · · · ·NEXT I
```

LOF(1)はファイル番号1のファイルの長さ（バイト数）、LEN(DENWA)は1レコードの長さ（バイト数）、¥は整数の割り算です。LOF(1) ¥ LEN(DENWA)はレコードの数を計算しているので、このループはレコードの数だけ繰り返すことになります。

GET #1, I, DENWAは、ランダムアクセスファイル#1のレコード番号I番のレコードを読みます。ファイルから読むときはDENWAでよいのですが、画面表示のときは、DENWA.NAMAE;　DENWA.BANGOU;　DENWA.S;　のように項目別に書きます。

次に、ランダムアクセスファイルにデータを追加入力するプログラムを作ります。このプログラムの構造は次のとおりです。




```
'電話帳6　　ランダムアクセスファイルへの追加の書き込み
DEFINT I                                          'レコード番号
TYPE DENWATYPE                                    'DENWATYPEというタイプを決める
     NAMAE  AS STRING * 14                        '名前は14桁（全角7字）
     BANGOU AS STRING * 16                        '電話番号は16桁（全角8字）
     S         AS INTEGER                         'レコードの削除のシルシ
END TYPE
DIM DENWA AS DENWATYPE                            'レコードのタイプはDENWATYPE

CLS                                               '画面消去
OPEN "B:DENWA.DAT" FOR RANDOM AS #1 LEN = LEN(DENWA)        '電話帳を開く
I = LOF(1) ¥ LEN(DENWA)                           'レコードの数を計算する
DENWA.S = -1                                      'レコードは有効
DO                                                '人数分繰り返す
     INPUT "名前(終了:E)：", N$                    '名前の入力(終了時はE)
     IF N$ = "E" THEN                             '　名前がEならば
          CLOSE #1                                '電話帳を閉じて
          END                                     '終了
     END IF
     DENWA.NAMAE = N$                             'N$を名前へコピー
     INPUT "電話番号：", DENWA.BANGOU               '電話番号の入力
     I = I + 1                                    'レコード番号へ1を加える
     PUT #1, I, DENWA                             '電話帳への書き込み
LOOP
END
```

```
I = LOF(1) ¥ LEN(DENWA)
```

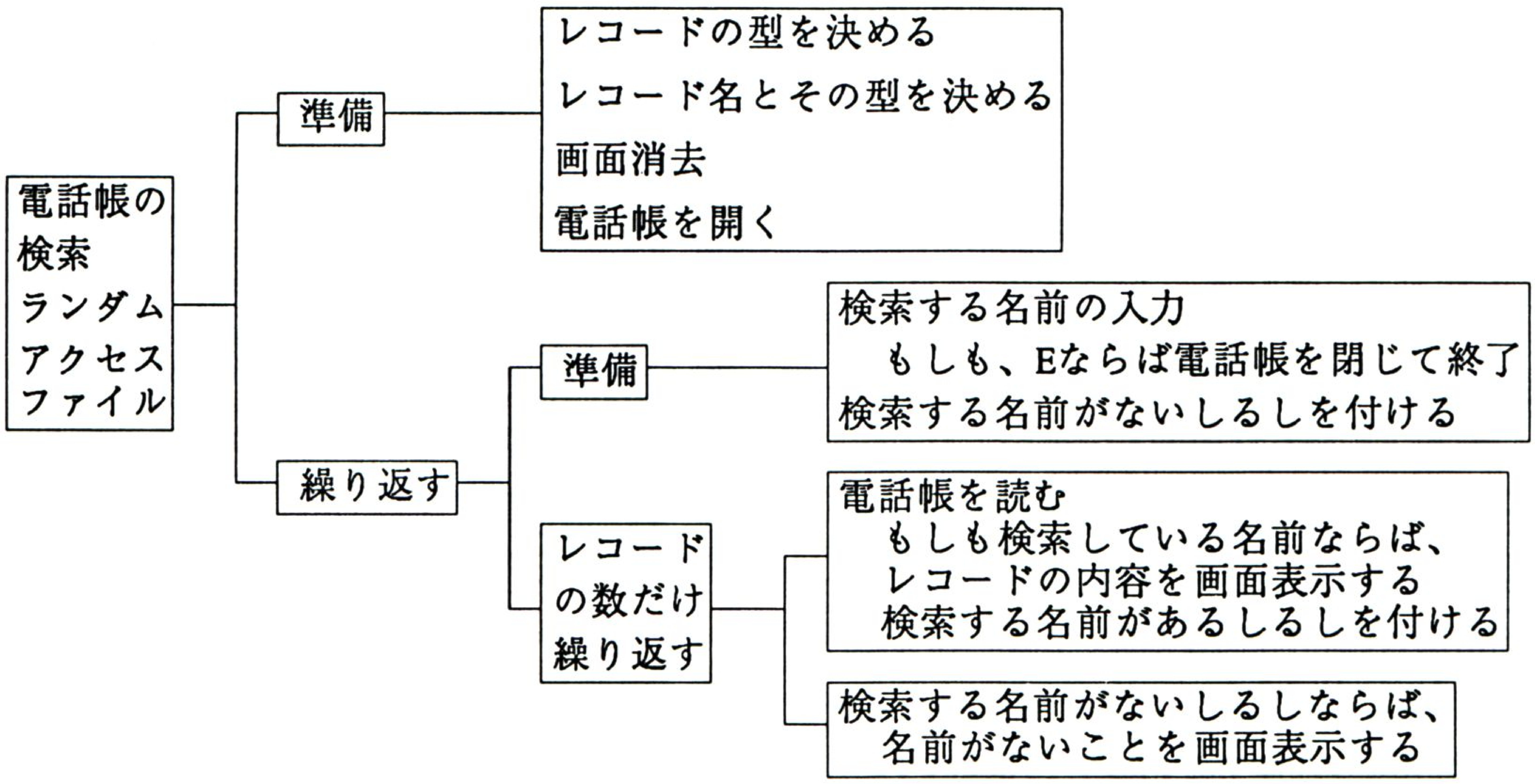
追加入力なので、すでに存在するレコードの数を計算してIに代入します。そして I = I + 1で追加のレコード番号を計算して、PUTステートメントで書き込みます。

5．3　電話番号の検索と訂正

電話帳は、電話番号を知るためにあります。そこで、名前を入力すると、電話番号が表示されるプログラムを作ります。

このような仕事を一般には検索といいます。名前が明確でない場合もあるので、例えば佐藤の佐や藤のみを入力しても、検索できるようにします。

このプログラムの構造は次のとおりです。




```
'電話帳7　　ランダムアクセスファイルの検索
DEFINT A, I                                             '整数型変数
TYPE DENWATYPE                                          'DENWATYPEというタイプを決める
    NAMAE  AS STRING * 14                               '名前は14桁（全角7字）
    BANGOU AS STRING * 16                               '電話番号は16桁（全角8字）
    S        AS INTEGER                                 'レコードの削除のシルシ
END TYPE
DIM DENWA AS DENWATYPE                                  'レコードのタイプはDENWATYPE
CLS                                                     '画面消去
OPEN "B:DENWA.DAT" FOR RANDOM AS #1 LEN = LEN(DENWA)            '電話帳を開く
DO
    INPUT "検索したい名前を入力してください。（一部も可、終了:E）",
                                                        KENSAKUGO$        '検索語の入力
    IF KENSAKUGO$ = "E" THEN                            'もしも、検索語がEならば
        CLOSE #1                                        '電話帳を閉じて
        END                                             '終了する
    END IF
    A = 0                                               '検索している名前がないしるし
    FOR I = 1 TO LOF(1) ¥ LEN(DENWA)                    'レコードの数だけ繰り返す
        GET #1, I, DENWA                                '読む
```

```
        IF KINSTR(DENWA.NAMAE, KENSAKUGO$) > 0 AND DENWA.S THEN
                    'もしも、名前に検索語があって、それが有効なレコードならば
            PRINT I, DENWA.NAMAE; DENWA.BANGOU        'レコードを画面表示する
            A = 1                                     '検索している名前があるしるし
        END IF
    NEXT I
    IF A = 0 THEN                         'もしも、検索している名前がないならば
        PRINT "該当する名前はありません。"          'ないことを画面表示する
    END IF
LOOP
END
```

```
IF KINSTR(DENWA. NAMAE,  KENSAKUGO$) > 0 AND DENWA. S THEN
```

KINSTR(DENWA.NAMAE,KENSAKUGO$)は、DENWA.NAMAEの文字列の中に検索のために入力したKENSAKUGO$があれば、その文字の位置を計算します。例えば、DENWA.NAMAEが佐々木　幸一郎でKENSAKUGO$が木ならば、A = KINSTR(DENWA.NAMAE, KENSAKUGO$)の計算結果は3になります。すなわち、木は佐々木の3番目の文字だからです。KINSTR関数は漢字（２バイト文字）専用の関数で、半角の英数の文字（１バイトの文字）にはINSTR関数を使います。

IF KINSTR(DENWA.NAMAE, KENSAKUGO$) > 0 は、KINSTR関数によって文字があればプラスの整数が、文字がなければゼロが計算されるので、「もしも、検索している文字があれば」ということになります。

ANDは、ANDの前後の条件が同時に真（同時に成立する）ならば、という意味です。

DENWA.Sは条件のように見えないのですが、パソコンは-1ならば真（成立する）と解釈しますから、これだけでDENWA.Sが真ならば、という条件になります。DENWA.Sにはあらかじめ-1（マイナス１）を入れておいて、真であればこのレコードが有効である（削除されていない）と判断させます。したがって、削除をするときには、DENWA.Sに0を入力しておくわけです。

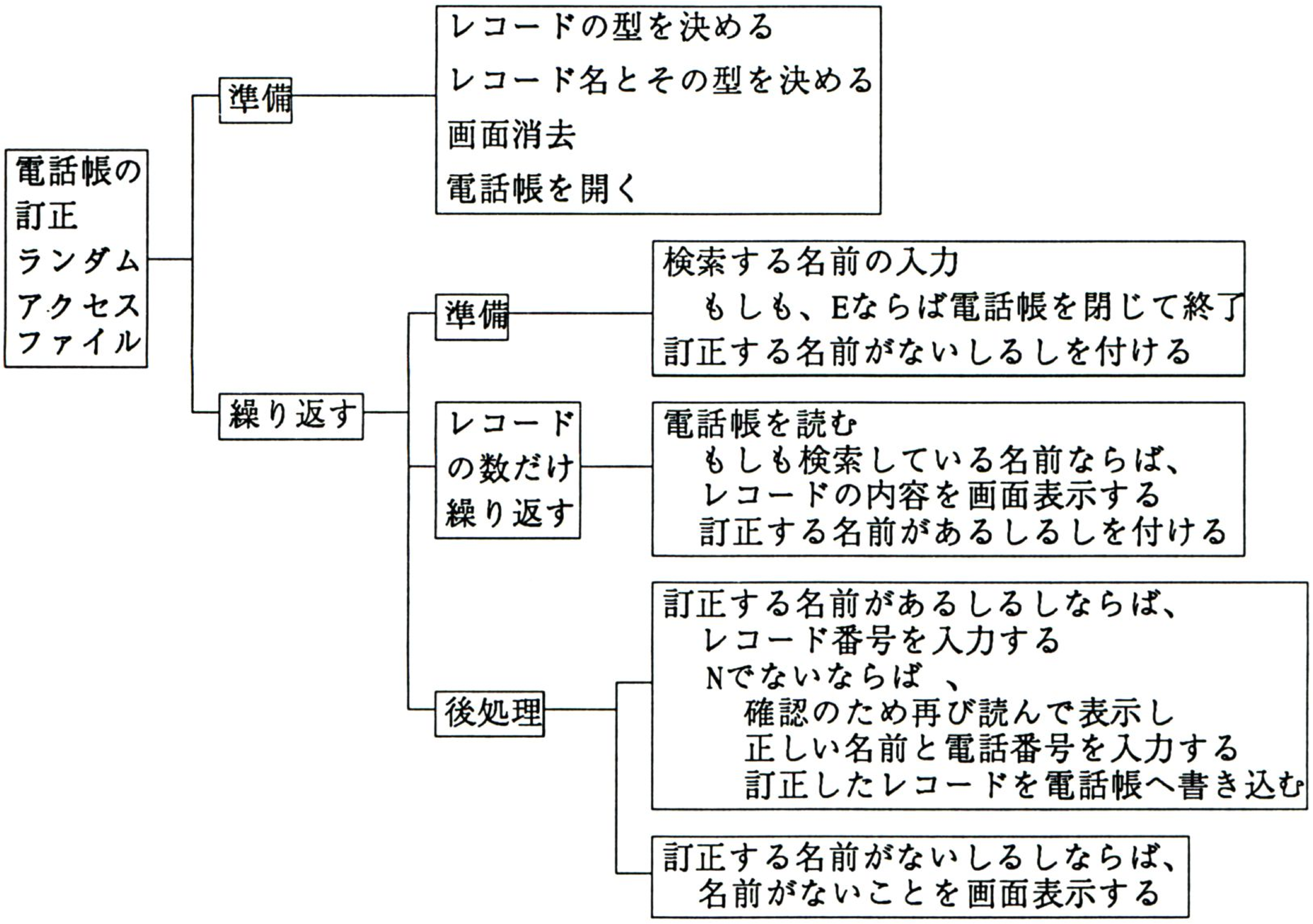
画面表示の例

```
検索したい名前を入力してください。（一部も可、終了:E）　藤
1 佐藤　花子　　　４５－７８９４
2 近藤　幸枝　　　５６－８５５２
検索したい名前を入力してください。（一部も可、終了:E）　上田
検索している名前はありません。
検索したい名前を入力してください。（一部も可、終了:E）　E
```

次に、ランダムアクセスファイルの訂正のプログラムを作ります。

訂正は、１レコードずつ点検しながら行う方法や、すべてのレコードの一覧表から一括して訂正を入力する方法などありますが、ここでは訂正すべきレコードを検索して訂正する方法で行います。

プログラムの構造は次のとおりです。




```
'電話帳８　ランダムアクセスファイルの訂正
DEFINT A, I, T                                          '整数型変数
TYPE DENWATYPE                                          'DENWATYPEというタイプを決める
    NAMAE  AS STRING * 14                               '名前は14桁（全角7字）
    BANGOU AS STRING * 16                               '電話番号は16桁（全角8字）
    S      AS INTEGER                                   'レコードの削除のシルシ
END TYPE
DIM DENWA AS DENWATYPE                                  'レコードのタイプはDENWATYPE
CLS                                                     '画面消去
OPEN "B:DENWA.DAT" FOR RANDOM AS #1 LEN = LEN(DENWA)              '電話帳を開く
DO
    INPUT "訂正したい名前を入力してください。（一部も可、終了:E）",
                                                        KENSAKUGO$          '検索語の入力
    IF KENSAKUGO$ = "E" THEN                            'もしも、検索語がEならば
        CLOSE #1                                        '電話帳を閉じて
        END                                             '終了する
    END IF
    A = 0                                               '該当の名前がないしるし
```

```
    FOR I = 1 TO LOF(1) ¥ LEN(DENWA)              'レコードの数だけ繰り返す
        GET #1, I, DENWA                          '読む
        IF KINSTR(DENWA.NAMAE, KENSAKUGO$) > 0 AND DENWA.S THEN
                    'もしも、名前に検索語があって、それが有効なレコードなら
            PRINT I; DENWA.NAMAE; DENWA.BANGOU    'レコードを画面表示する
            A = 1                                 '訂正する名前があるしるし
        END IF
    NEXT I
    SELECT CASE A
        CASE 1
          INPUT"訂正するレコード番号を入力してください。(訂正しない:N)",
                                                  TEISEI$
            IF TEISEI$ <> "N" THEN                '番号がNでないならば
                TEISEI = VAL(TEISEI$)             'TEISEI$を整数に変えて
                GET #1, TEISEI, DENWA            '念のため訂正するレコードを読む
                PRINT DENWA.NAMAE; "--->";        '訂正する名前を表示し
                INPUT DENWA.NAMAE                 '正しい名前を入力
                PRINT DENWA.BANGOU; "--->";       '訂正する電話番号を表示し
                INPUT DENWA.BANGOU                '正しい電話番号を入力
                PUT #1, TEISEI, DENWA             '電話帳へ書き込む
            END IF
        CASE 0                              'もしも、訂正する名前がないならば
            PRINT "訂正する名前はありません。"         'ないことを画面表示する
    END SELECT
LOOP
END
```

画面表示の例

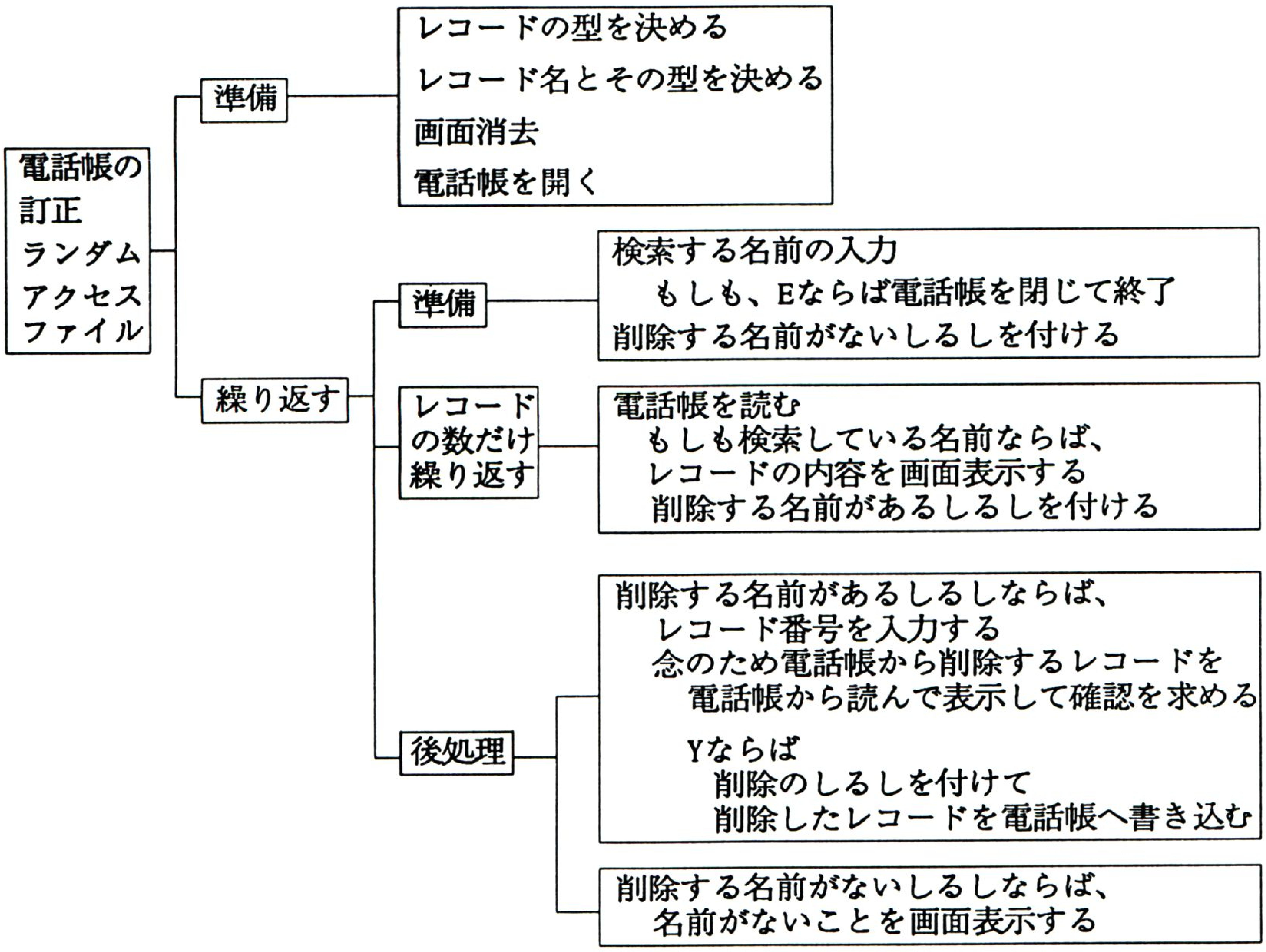
```
訂正したい名前を入力してください。(一部も可、終了:E)　藤
1 佐藤　花子　　　45-7894
5 近藤　幸枝　　　56-8552
訂正するレコード番号を入力してください。(訂正しない:N)　5
近藤　幸枝　　--->? 近藤　恵子
56-8552　--->? 56-8553
訂正したい名前を入力してください。(一部も可、終了:E)　上田
訂正する名前はありません。
訂正したい名前を入力してください。(一部も可、終了:E)　E
```

5．4　電話番号の削除と復活

次に、ランダムアクセスファイルのレコードを削除するプログラムを作ります。

ただし、前節でも触れたように、実際には削除しないで削除のしるしを付けておく方法です。前節と同様に、削除するレコードを検索してから削除します。

プログラムの構造は次のとおりです。




```
'電話帳9　　ランダムアクセスファイルの削除
DEFINT I,S-T                                          '整数型変数
TYPE DENWATYPE                                        'DENWATYPEというタイプを決める
    NAMAE AS STRING * 14                              '名前は14桁（全角7字）
    BANGOU AS STRING * 16                             '電話番号は16桁（全角8字）
    S         AS INTEGER                              'レコードの削除のシルシ
END TYPE
DIM DENWA AS DENWATYPE                                'レコードのタイプはDENWATYPE

CLS                                                   '画面消去
OPEN "B:DENWA.DAT" FOR RANDOM AS #1 LEN = LEN(DENWA)          '電話帳を開く
DO
    INPUT "削除したい名前を入力してください。（一部も可、終了:E）",
                                          KENSAKUGO$          '検索語の入力
```

```
    IF KENSAKUGO$ = "E" THEN                    'もしも、検索語がEならば
        CLOSE #1                                '電話帳を閉じて
        END                                     '終了する
    END IF
    A = 0                                       '削除したい名前がないしるし
    FOR I = 1 TO LOF(1) ¥ LEN(DENWA)            'レコードの数だけ繰り返す
        GET #1, I, DENWA                        '読む
        IF KINSTR(DENWA.NAMAE, KENSAKUGO$) > 0 AND DENWA.S THEN
                '↑もしも、名前に検索語があって、それが有効なレコードならば
            PRINT I; DENWA.NAMAE; DENWA.BANGOU   'レコードを画面表示する
            A = 1                               '削除したい名前があるしるし
        END IF
    NEXT I
    SELECT CASE A
        CASE 1                                  'もしも検索した名前があれば
          INPUT"削除するレコード番号を入力してください。(削除しない:N) ",
            SAKUZYO$                            '削除するレコード番号の入力
            IF SAKUZYO$ <> "N" THEN             'レコード番号がNでないならば
                SAKUZYO = VAL(SAKUZYO$)         '削除番号を整数にする
                GET #1, SAKUZYO, DENWA          '削除するレコードを読む
                PRINT SAKUZYO, DENWA. NAMAE; DENWA. BANGOU
                                                '確認のためレコードを表示する
                PRINT
          "**確認**このレコードを削除してよろしいですか。はい:Y いいえ:N"
                                            '本当に削除してよいか確認を求める
                DO
                    KAKUNIN$ = INKEY$           '確認のためYかNを入力
                LOOP UNTIL KAKUNIN$ = "Y" OR KAKUNIN$ = "N"
                                                'もしもY,N以外なら繰り返す
                IF KAKUNIN$ = "Y" THEN          'もしも、Yならば
                    DENWA.S = 0                 '削除のしるしを付けて
                    PUT #1, SAKUZYO, DENWA      '再度書き込む
                END IF
            END IF
        CASE 0                                  'もしも検索した名前がなければ
            PRINT "削除する名前はありません。"    'ないことを画面表示する
    END SELECT
LOOP
END
```

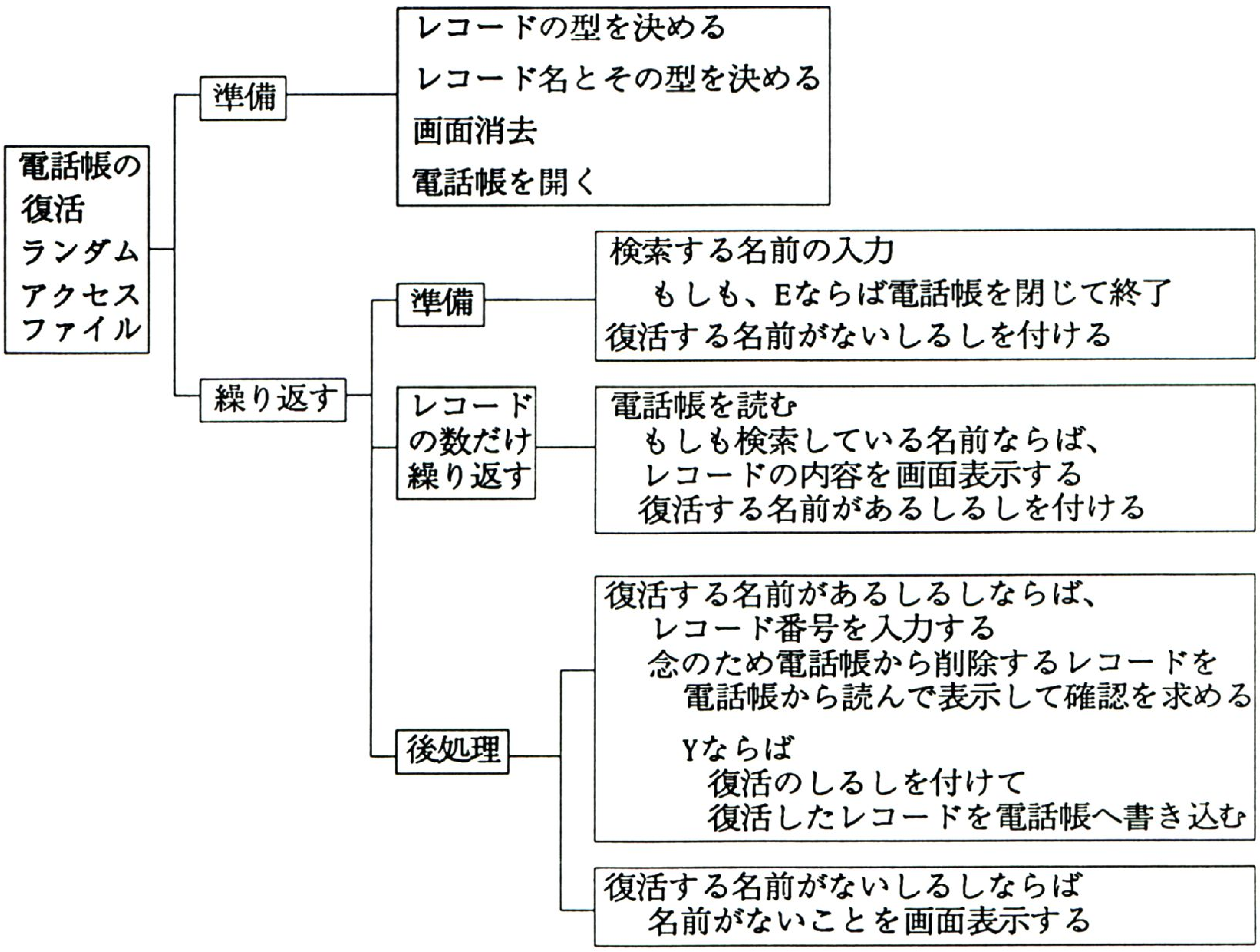
削除のプログラムは、本当に削除するかどうかの確認を２度行っています。

レコード番号を入力するところでNを押すと削除しないことになります。また、レコード番号を入力した後にも、再度レコードを表示して、YかNを入力して確認するようにしています。

画面表示の例

```
削除したい名前を入力してください。（一部も可、終了:E）　藤
1 佐藤　花子　　　　４５－７８９４
5 近藤　恵子　　　　５６－８５５３
削除するレコード番号を入力してください。（削除しない:N）　5
5 近藤　恵子　　　　５６－８５５３
**確認**このレコードを削除してよろしいですか。はい:Y　いいえ:N
削除したい名前を入力してください。（一部も可、終了:E）　E
```

次に、削除したレコードを復活するプログラムを作ります。
プログラムの構造は次のとおりです。




```
'電話帳１０　　ランダムアクセスファイルの復活
DEFINT H-I, T                          '整数型変数
TYPE DENWATYPE                         'DENWATYPEというタイプを決める
     NAMAE AS STRING * 14              '名前は14桁（全角7字）
     BANGOU AS STRING * 16             '電話番号は16桁（全角8字）
```

```
    S          AS INTEGER                         'レコードの削除のシルシ
END TYPE
DIM DENWA AS DENWATYPE                            'レコードのタイプはDENWATYPE

CLS                                               '画面消去
OPEN "B:DENWA.DAT" FOR RANDOM AS #1 LEN = LEN(DENWA)           '電話帳を開く
DO
    INPUT "復活したい名前を入力してください。(一部も可、終了:E)    ",
                                                  KENSAKUGO$        '検索語の入力
    IF KENSAKUGO$ = "E" THEN                      'もしも、検索語がEならば
        CLOSE #1                                  '電話帳を閉じて
        END                                       '終了する
    END IF
    A = 0                                         '該当の名前がないしるし
    FOR I = 1 TO LOF(1) ¥ LEN(DENWA)              'レコードの数だけ繰り返す
        GET #1, I, DENWA                          '読む
        IF KINSTR(DENWA.NAMAE, KENSAKUGO$) > 0 AND NOT DENWA.S THEN
                    'もしも、名前に検索語があって、それが有効なレコードならば
            PRINT I; DENWA.NAMAE; DENWA.BANGOU       'レコードを画面表示する
            A = 1                                 '復活する名前があるしるし
        END IF
    NEXT I
    SELECT CASE A
        CASE 1                                    'もしも検索した名前があれば
          INPUT"復活するレコード番号を入力してください。(復活しない:N)",
            HUKKATU$                              '復活するレコード番号の入力
            IF HUKKATU$ <> "N" THEN               'レコード番号がNでないならば
                HUKKATU = VAL(HUKKATU$)           '復活番号を整数にする
                GET #1, HUKKATU, DENWA            '復活するレコードを読む
                PRINT HUKKATU; DENWA. NAMAE; DENWA. BANGOU
                                                  '確認のためレコードを表示する
                PRINT
          "**確認**このレコードを復活してよろしいですか。はい:Y  いいえ:N"
                                              '本当に復活してよいか確認を求める
                DO
                    KAKUNIN$ = INKEY$             '確認のためYかNを入力
                LOOP UNTIL KAKUNIN$ = "Y" OR KAKUNIN$ = "N"
                                                  'もしもY,N以外なら繰り返す
                IF KAKUNIN$ = "Y" THEN            'もしも、Yならば
                    DENWA.S = -1                  '復活のしるしを付けて
                    PUT #1, HUKKATU, DENWA        '再度書き込む
                END IF
            END IF
        CASE 0                                    'もしも検索した名前がなければ
            PRINT "復活する名前はありません。"       'ないことを画面表示する
    END SELECT
LOOP
END
```

```
IF KINSTR(DENWA.NAMAE, KENSAKUGO$) > 0 AND NOT DENWA.S THEN
```

NOT DENWA.S は、「DENWA.Sが偽ならば」ということです。レコードを削除するときにはDENWA.Sを0（ゼロ）にしていますから、言い替えれば「すでに削除されたレコードならば」ということになります。DENWA.S = 0 としても同じです。

復活をさせる場合は、DENWA.S = -1 として電話帳へ書き込んでいます。

画面表示の例

```
復活したい名前を入力してください。（一部も可、終了:E）　藤
5 近藤　恵子　　　　５６－８５５３
復活するレコード番号を入力してください。（復活しない:N）　5
5 近藤　恵子　　　　５６－８５５３
**確認**このレコードを復活してよろしいですか。はい:Y　いいえ:N
復活したい名前を入力してください。（一部も可、終了:E）　E
```

5．5　漢字入力モード

漢字を入力するときには、これまでは［CTRL］と［XFER］を同時に押して、漢字入力モードに変えてから漢字の入力を行ってきました。また、漢字入力モードから英数モードへ戻すときも、同様に［CTRL］と［XFER］を同時に押さなければなりません。

これまで作ってきた電話帳では、このようにモードの変更をしていたのでは操作が面倒なので、名前とともに電話番号も全角の漢字で入力して、最後に終了のためにEを入力するときだけ、英数モードに戻してきました。

この漢字入力モードの変更を必要に応じて自動的に行うようにすれば、キーボードからの操作が不要となり、入力が快適になります。

モード変更を自動的に行うためのプログラムを作る手順は、次のとおりです。

1. まず、Ｑｕｉｃｋ　ＢＡＳＩＣの起動の際に、A>QBと入力していたところを、今度は、A>QB　/LGEN　と入力して起動します。

これは、AドライブにあるＱＢ起動ディスクの中のLIBというサブディレクトリに存在するＱｕｉｃｋライブラリのGEN.QLBを、Ｑｕｉｃｋ　ＢＡＳＩＣの環境下で利用するために起動時にメモリーに読み込むことを求めているのです。

GEN.QLBは、Ｑｕｉｃｋ　ＢＡＳＩＣがもっていない機能を、拡張ライブラリとして提供したものの一つです。

また、別にGEN.LIBというライブラリもあります。これは、実行用のプログラムを作る際に、LINKによって結合されるライブラリです。

```
A>QB /LGEN
```

2. プログラムの先頭部分に、次のように書き加えます。

```
'$INCLUDE: 'QB.BI'
'$INCLUDE: 'GEN.BI'
```

QB.BI及びGEN.BIは、ＱＢワークディスクのINCLUDEディレクトリに存在します。これらをインクルードファイルといい、Ｑｕｉｃｋ　ＢＡＳＩＣに対してライブラリの中の命令の仕様を知らせる仕事をします。

3. ライブラリのサブプロシージャを使います。

漢字入力の直前に**Fep "ON"** とし、漢字入力の後に**Fep "OFF"** とします。

```
Fep "ON"                    '漢字入力のモードへ変える
     INPUT （漢字データ）   '漢字のデータを入力する
Fep "OFF"                   '英数モードへ変える
```

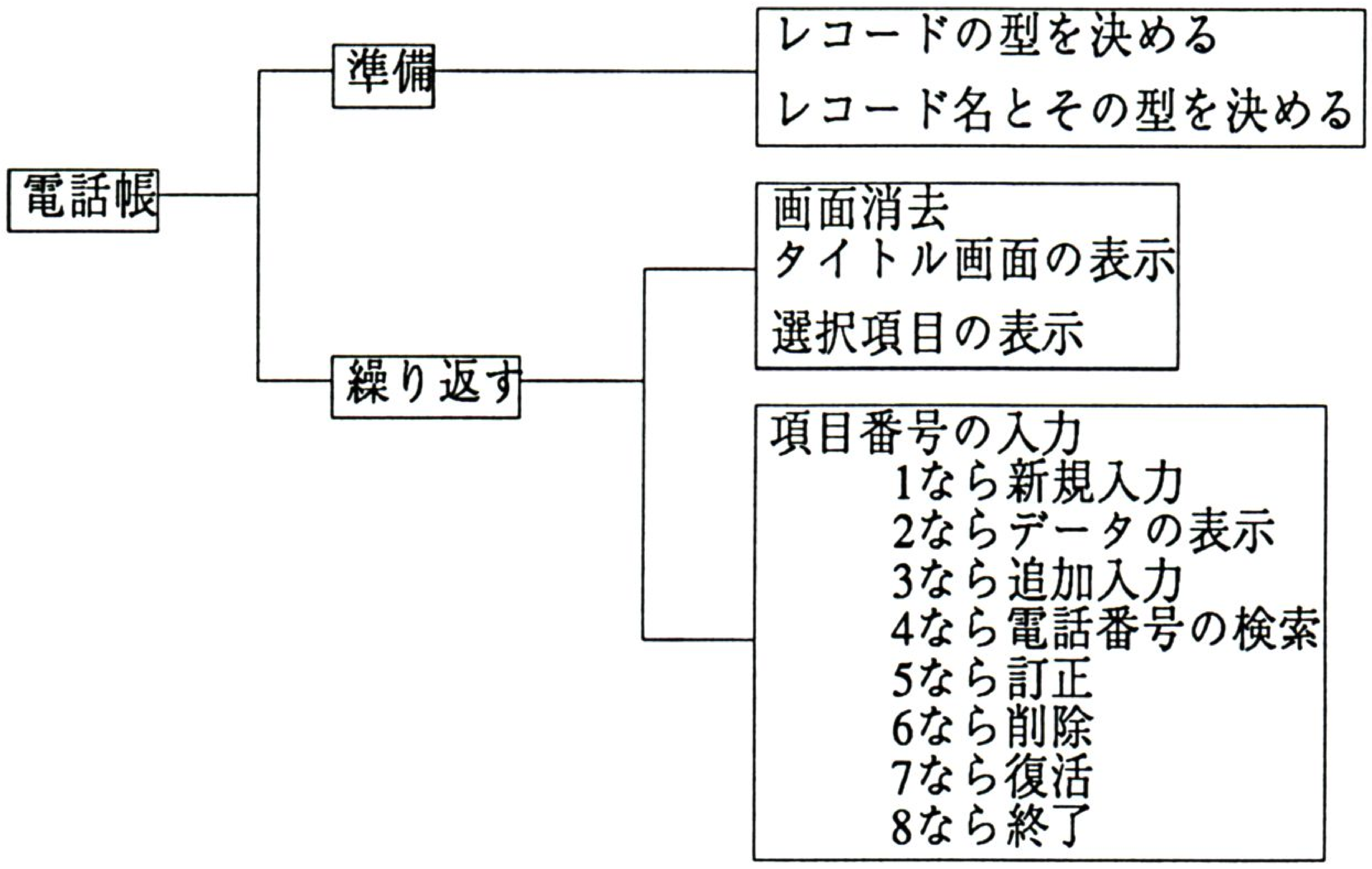
サンプルプログラムは次のとおりです。

```
'漢字モード変更のサンプル
'$INCLUDE: 'QB.BI'                        'インクルードファイル
'$INCLUDE: 'GEN.BI'
DEFINT B
DO                                        '繰り返す
     Fep "ON"                             '漢字モードにする
     INPUT "名前：", NAMAE$               '名前の入力
     Fep "OFF"                            '英数モードにする
     IF NAMAE$ = "E" THEN                 'NAMAE$がEならば終了
          END
     END IF
     INPUT "番号（半角）：", BANGOU       '半角の番号を入力する
     PRINT NAMAE$; BANGOU                 '名前と番号を画面表示する
LOOP
```

このように、漢字入力モードを自動的に切り換える方法によって電話帳の入力を行えば、電話番号を半角の数字としても、容易に入力できるようになります。

５．６　電話帳のまとめ

いよいよ、電話帳のプログラムをまとめる段階になりました。
電話帳全体のプログラムの構造は次のとおりです。



これまで作ってきたプログラムをまとめる手順はつぎのとおりです。

1. はじめにタイトル画面と項目を選択するメインプロシージャを作ります。

2. SUB(ｻﾌﾞﾌﾟﾛｼｰｼﾞｬ名)と入力すると、画面がサブプロシージャの画面に変わり、自動的にEND SUBも生成されます。

3. SUB(ｻﾌﾞﾌﾟﾛｼｰｼﾞｬ名)とEND SUBとの間にカーソルを置いて、次の操作をすると、プログラムがカーソルの位置に読み込まれます。
　ただし、読み込まれるプログラムは、保存するときにテキストファイルの形式で保存されたものであることが必要です。もしも読み込めないときは、メインプロシージャを保存してから、これまでのプログラムを読み込んで再度テキストファイルの形式で保存しなおしてください。

1.マウスカーソルをF/ﾌｧｲﾙに合わせて左クリック
2.マウスカーソルをM/結合 ... に合わせて左クリック
3.マウスカーソルをD/ディレクトリ欄の-B-に合わせて左ダブルクリック
4.マウスカーソルをF/ファイル欄のプログラム名
　（例えば検索ならDENWA7.BAS）に合わせて左ダブルクリック

なお、M/結合 ... のメニューが表示されない場合は、O/ｵﾌﾟｼｮﾝ　F/FULLﾒﾆｭｰを左クリックしておいて、再度　F/ﾌｧｲﾙ　のクリックから操作しなおしてください。

4. すべてのサブプロシージャを作ったら、調整作業をします。

1.TYPE DENWATYPE....END TYPE はメインプロシージャにおくだけにして、サブのTYPE.....END TYPEは削除します。

2.DEFINTは第１行目にあるものを使うので、読み込んだ場合は削除します。

3.ENDがサブプロシージャの最後の行にあるEND SUBの上に位置している場合は、END SUBがあるので削除します。

4.サブプロシージャの中で、Eを判定して終了する場合があります。そこにはCLOSE #1とENDが続いていますが、そのままではこの位置で終了しますので、ここからメインプロシージャへ戻るように、ENDをEXIT SUBに替えます。

修正前

```
IF  .......   ="E" THEN
      CLOSE #1
      END
END IF
```

修正後

```
IF  .......   ="E" THEN
      CLOSE #1
      EXIT SUB
END IF
```

5. 漢字入力モードのＯＮとＯＦＦの動作を自動的に行うため、名前のINPUTの直前に FEP "ON" を、直後には FEP "OFF"を挿入します。

［メインプロシージャ］

```
DECLARE SUB TEISEIS ()
DECLARE SUB SINKI ()
DECLARE SUB HYOUZI ()
DECLARE SUB TUIKA ()
DECLARE SUB KENSAKU ()
DECLARE SUB SAKUZYOS ()
DECLARE SUB HUKKATUS ()
'電話帳１１　　まとめ
'$INCLUDE: 'QB.BI'                              'インクルードファイル
'$INCLUDE: 'GEN.BI'
TYPE DENWATYPE                                  '型の設定
    NAMAE  AS STRING * 14                       '名前は１４バイト
    BANGOU AS STRING * 16                       '番号は１６バイト
    S      AS INTEGER                    'レコードの削除のしるし：整数２バイト
END TYPE
DEFINT A, H-I, S-T                              '整数型変数
DIM DENWA AS DENWATYPE                          'DENWAレコードはDENWATYPE型
DO                                              '繰り返す
    CLS                                         '画面表示
    COLOR 14:LOCATE 1, 25: PRINT "                             "
             LOCATE 2, 25: PRINT "                             "
    COLOR 11:LOCATE 3, 25: PRINT "      あなたの電話帳          "
    COLOR 14:LOCATE 4, 25: PRINT "                             "
             LOCATE 5, 25: PRINT "                             "
    COLOR 2: LOCATE 7, 30: PRINT "1  新規の入力"
             LOCATE 9, 30: PRINT "2  データの表示"
             LOCATE 11,30: PRINT "3  データの追加入力"
             LOCATE 13,30: PRINT "4  電話番号の検索"
             LOCATE 15,30: PRINT "5  データの訂正"
             LOCATE 17,30: PRINT "6  データの削除"
             LOCATE 19,30: PRINT "7  データの復活"
             LOCATE 21,30: PRINT "8  終    了"
    COLOR 14:LOCATE 23,25: PRINT "選択して番号を押してください。"
    COLOR 7
    DO                                          '１文字入力
        I$ = INKEY$
    LOOP UNTIL I$ <> ""                         '何か押されるまで繰り返す
    SELECT CASE I$
        CASE "1": SINKI                         '1なら新規入力
        CASE "2": HYOUZI                        '2なら表示
        CASE "3": TUIKA                         '3なら追加
        CASE "4": KENSAKU                       '4なら検索
        CASE "5": TEISEIS                       '5なら訂正
        CASE "6": SAKUZYOS                      '6なら削除
        CASE "7": HUKKATUS                      '7なら復活
        CASE "8": END                           '8なら終了
    END SELECT
LOOP
```

[新規入力のサブプロシージャ]

```
SUB SINKI
'電話帳4　ランダムアクセスファイルの新規入力
DIM DENWA AS DENWATYPE                          'レコードのタイプはDENWATYPE

OPEN "B:DENWA.DAT" FOR RANDOM AS #1 LEN = LEN(DENWA)        '電話帳を開く
I =0                                            'レコード番号を0にする
DENWA.S = -1                                    'レコードは有効
DO                                              '人数分繰り返す
    FEP "ON"                                    '漢字入力モードＯＮ
    INPUT "名前(終了:E)：", N$                  '名前の入力(終了時はE)
    FEP "OFF"                                   '漢字入力モードＯＦＦ
    IF N$ = "E" THEN                            '名前がEならば
        CLOSE #1                                '電話帳を閉じて
        EXIT SUB                                '終了
    END IF
    DENWA.NAMAE = N$                            'N$を電話番号へコピー
    INPUT "電話番号：", DENWA.BANGOU            '電話番号の入力
    I =I +1                                     'レコード番号へ1を加える
    PUT #1, I, DENWA                            '電話帳への書き込み
LOOP
END SUB
```

[表示のサブプロシージャ]

```
SUB HYOUZI
'電話帳5　ランダムアクセスファイルを読んで画面表示
DIM DENWA AS DENWATYPE                          'レコードのタイプはDENWATYPE
CLS                                             '画面消去
OPEN "B:DENWA.DAT" FOR RANDOM AS #1 LEN = LEN(DENWA)        '電話帳を開く
FOR I = 1 TO LOF(1) ¥ LEN(DENWA)                'レコードの数だけ繰り返す
    GET #1, I, DENWA                            '読む
    IF DENWA.S THEN
        PRINT I; DENWA.NAMAE; DENWA.BANGOU; DENWA.S         '画面へ表示
    END IF
NEXT I
CLOSE #1
PRINT "何かキーを押してください。"
DO
    LOOP UNTIL INKEY$ <> ""
END SUB
```

［追加入力のサブプロシージャ］

```
SUB TUIKA
'電話帳6　　ランダムアクセスファイルへの追加の書き込み
DIM DENWA AS DENWATYPE                                  'レコードのタイプはDENWATYPE
CLS                                                     '画面消去
OPEN "B:DENWA.DAT" FOR RANDOM AS #1 LEN = LEN(DENWA)    '電話帳を開く
I = LOF(1) ¥ LEN(DENWA)                                 'レコードの数を計算する
DENWA.S = -1                                            'レコードは有効
DO                                                      '人数分繰り返す
    FEP "ON"                                            '漢字入力モードON
    INPUT "名前(終了:E)：", N$                          '名前の入力(終了時はE)
    FEP "OFF"                                           '漢字入力モードOFF
    IF N$ = "E" THEN                                    '名前がEならば
        CLOSE #1                                        '電話帳を閉じて
        EXIT SUB                                        '終了
    END IF
    DENWA.NAMAE = N$                                    'N$を名前へコピー
    INPUT "電話番号：", DENWA.BANGOU                    '電話番号の入力
    I = I + 1                                           'レコード番号へ1を加える
    PUT #1, I, DENWA                                    '電話帳への書き込み
LOOP
END SUB
```

［電話番号の検索のサブプロシージャ］

```
SUB KENSAKU
'電話帳7　　ランダムアクセスファイルの検索
DIM DENWA AS DENWATYPE                                  'レコードのタイプはDENWATYPE
CLS                                                     '画面消去
OPEN "B:DENWA.DAT" FOR RANDOM AS #1 LEN = LEN(DENWA)    '電話帳を開く
DO
    FEP "ON"                                            '漢字入力モードON
    INPUT "検索したい名前を入力してください。（一部も可、終了:E）　",
                                                        KENSAKUGO$    '検索語の入力
    FEP "OFF"                                           '漢字入力モードOFF
    IF KENSAKUGO$ = "E" THEN                            'もしも、検索語がEならば
        CLOSE #1                                        '電話帳を閉じて
        EXIT SUB                                        '終了する
    END IF
    A = 0                                               '検索する名前がないしるし
    FOR I = 1 TO LOF(1) ¥ LEN(DENWA)                    'レコードの数だけ繰り返す
        GET #1, I, DENWA                                '読む
        IF KINSTR(DENWA.NAMAE, KENSAKUGO$) > 0 AND DENWA.S THEN
                        'もしも、名前に検索語があって、それが有効なレコードならば
            PRINT I; DENWA.NAMAE; DENWA.BANGOU          'レコードを画面表示する
            A = 1                                       '検索する名前があるしるし
        END IF
    NEXT I
    IF A = 0 THEN                                       'もしも、検索する名前がないならば
        PRINT "検索している名前はありません。"          'ないことを画面表示する
```

```
    END IF
LOOP
END SUB
```

[訂正のサブプロシージャ]

```
SUB TEISEIS
'電話帳８　ランダムアクセスファイルの訂正
DIM DENWA AS DENWATYPE                          'レコードのタイプはDENWATYPE
CLS                                             '画面消去
OPEN "B:DENWA.DAT" FOR RANDOM AS #1 LEN = LEN(DENWA)       '電話帳を開く
DO
    FEP "ON"                                    '漢字入力モードON
    INPUT "訂正したい名前を入力してください。（一部も可、終了:E）  ",
                                                KENSAKUGO$     '検索語の入力
    FEP "OFF"                                   '漢字入力モードOFF
    IF KENSAKUGO$ = "E" THEN                    'もしも、検索語がEならば
        CLOSE #1                                '電話帳を閉じて
        EXIT SUB                                '終了する
    END IF
    A = 0                                       '訂正する名前がないしるし
    FOR I = 1 TO LOF(1) ¥ LEN(DENWA)            'レコードの数だけ繰り返す
        GET #1, I, DENWA                        '読む
        IF KINSTR(DENWA.NAMAE, KENSAKUGO$) > 0 AND DENWA.S THEN
                      '↑もしも、名前に検索語があって、それが有効なレコードならば
            PRINT I; DENWA.NAMAE; DENWA.BANGOU  'レコードを画面表示する
            A = 1                               '訂正する名前があるしるし
        END IF
    NEXT I
    SELECT CASE A
        CASE 1
        INPUT "訂正するレコード番号を入力してください。（訂正しない:N）",
                                                TEISEI$
            IF TEISEI$ <> "N" THEN
                TEISEI = VAL(TEISEI$)
                GET #1, TEISEI, DENWA
                PRINT DENWA.NAMAE; "--->";
                FEP "ON"
                INPUT DENWA.NAMAE
                FEP "OFF"
                PRINT DENWA.BANGOU; "--->";
                INPUT DENWA.BANGOU
                PUT #1, TEISEI, DENWA
            END IF
        CASE 0                                  'もしも、訂正する名前がないならば
            PRINT "訂正する名前はありません。"     'ないことを画面表示する
    END SELECT
LOOP
END SUB
```

［削除のサブプロシージャ］

```
SAB SAKUZYOS
'電話帳9　ランダムアクセスファイルの削除
DIM DENWA AS DENWATYPE                              'レコードのタイプはDENWATYPE
CLS                                                 '画面消去
OPEN "B:DENWA.DAT" FOR RANDOM AS #1 LEN = LEN(DENWA)          '電話帳を開く
DO
    FEP "ON"                                        '漢字入力モードＯＮ
    INPUT "削除したい名前を入力してください。（一部も可、終了:E）  ",
                                                    KENSAKUGO$      '検索語の入力
    FEP "OFF"                                       '漢字入力モードＯＦＦ
    IF KENSAKUGO$ = "E" THEN                        'もしも、検索語がEならば
        CLOSE #1                                    '電話帳を閉じて
        EXIT SUB                                    '終了する
    END IF
    A = 0                                           '削除する名前がないしるし
    FOR I = 1 TO LOF(1) ¥ LEN(DENWA)                'レコードの数だけ繰り返す
        GET #1, I, DENWA                            '読む
        IF KINSTR(DENWA.NAMAE, KENSAKUGO$) > 0 AND DENWA.S THEN
                        'もしも、名前に検索語があって、それが有効なレコードならば
            PRINT I; DENWA.NAMAE; DENWA.BANGOU      'レコードを画面表示する
            A = 1                                   '該当する名前があるしるし
        END IF
    NEXT I
    SELECT CASE A
        CASE 1                                      'もしも検索した名前があれば
        INPUT "削除するレコード番号を入力してください。（削除しない:N）",
            SAKUZYO$                                '削除するレコード番号の入力
            IF SAKUZYO$ <> "N" THEN                 'レコード番号がNでないならば
                SAKUZYO = VAL(SAKUZYO$)             '削除番号を整数にする
                GET #1, SAKUZYO, DENWA              '削除するレコードを読む
                PRINT I; DENWA.NAMAE; DENWA.BANGOU
                                                '確認のためレコードを画面表示する
 PRINT "**確認**このレコードを削除してよろしいですか。はい:Y  いいえ:N  "
                                                '本当に削除してよいか確認を求める
                DO
                    KAKUNIN$ = INKEY$               '確認のためYかNを入力
                LOOP UNTIL KAKUNIN$ = "Y" OR KAKUNIN$ = "N"
                                                    'もしもY,N以外なら繰り返す
                IF KAKUNIN$ = "Y" THEN              'もしも、Yならば
                    DENWA.S = 0                     '削除のしるしを付けて
                    PUT #1, SAKUZYO, DENWA          '再度書き込む
                END IF
            END IF
        CASE 0                                      'もしも、検索した名前がないならば
            PRINT "削除する名前はありません。"       'ないことを画面表示する
    END SELECT
LOOP
END SUB
```

[復活のサブプロシージャ]

```
SUB HUKKATUS
'電話帳１０　ランダムアクセスファイルの復活
DIM DENWA AS DENWATYPE                              'レコードのタイプはDENWATYPE
CLS                                                 '画面消去
OPEN "B:DENWA.DAT" FOR RANDOM AS #1 LEN = LEN(DENWA)            '電話帳を開く
DO
    FEP "ON"                                        '漢字入力モードＯＮ
    INPUT "復活したい名前を入力してください。（一部も可、終了:E）   ",
                                                    KENSAKUGO$      '検索語の入力
    FEP "OFF"                                       '漢字入力モードＯＦＦ
    IF KENSAKUGO$ = "E" THEN                        'もしも、検索語がEならば
        CLOSE #1                                    '電話帳を閉じて
        EXIT SUB                                    '終了する
    END IF
    A = 0                                           '該当の名前がないしるし
    FOR I = 1 TO LOF(1) ¥ LEN(DENWA)                'レコードの数だけ繰り返す
        GET #1, I, DENWA                            '読む
        IF KINSTR(DENWA.NAMAE, KENSAKUGO$) > 0 AND NOT DENWA.S THEN
                        '↑もしも、名前に検索語があって、それが有効なレコードならば
            PRINT I; DENWA.NAMAE; DENWA.BANGOU      'レコードを画面表示する
            A = 1                                   '該当する名前があるしるし
        END IF
    NEXT I
    SELECT CASE A
        CASE 1                                      'もしも検索した名前があれば
        INPUT "復活するレコード番号を入力してください。（復活しない:N）",
                                                    HUKKATU$
                                                    '復活するレコード番号の入力
            IF HUKKATU$ <> "N" THEN                 'レコード番号がNでないならば
                HUKKATU = VAL(HUKKATU$)             '復活番号を整数にする
                GET #1, HUKKATU, DENWA              '復活するレコードを読む
                PRINT I; DENWA.NAMAE; DENWA.BANGOU
                                                '確認のためレコードを画面表示する
  PRINT "**確認**このレコードを復活してよろしいですか。はい:Y  いいえ:N  "
                                                '本当に復活してよいか確認を求める
                DO
                    KAKUNIN$ = INKEY$               '確認のためYかNを入力
                LOOP UNTIL KAKUNIN$ = "Y" OR KAKUNIN$ = "N"
                                                    'もしもY,N以外なら繰り返す
                IF KAKUNIN$ = "Y" THEN              'もしも、Yならば
                    DENWA.S = -1                    '復活のしるしを付けて
                    PUT #1, HUKKATU, DENWA          '再度書き込む
                END IF
            END IF
        CASE 0                                  'もしも、検索した名前がないならば
            PRINT "該当する名前はありません。"      'ないことを画面表示する
    END SELECT
LOOP
END SUB
```

第6章

音楽ソフト管理

プログラムの工夫とインデックスファイルについて学びます。

6．1　同じデータの入力

パソコンを利用した音楽ソフト管理について学びます。

音楽ソフトとは、カセットテープ、コンパクトディスク、レコード、レーザディスク、ビデオテープなどいろいろな媒体に記憶されている音楽のことです。中でもカセットテープやビデオテープは放送を録音・録画する場合もあるので、その保管や整理に困っている方も多いと思います。

まず、管理のためにどのようなデータを入力したらよいかを考えます。

1． 媒体

2桁のアルファベットの半角の記号とします。

カセットテープ:TP　レコード:RC　コンパクトディスク:CD

レーザディスク:LD　ビディオディスク:VD

2． 登録番号

半角の4桁の数字とします。2桁は年を表し、2桁は連番とします。

例えば、1990年の第1番に登録すれば、9001番とします。

3． 曲名

全角文字列14字（STRING * 28）とします。

4． 曲名の読み

全角のひらかな4文字とします。

5． 歌手名

全角文字列9字（STRING * 18）とします。

6． 歌手名の読み

全角のひらかな4文字とします。

入力するテスト用のデータは、次のとおりです。

媒体	登録番号	曲　　名	曲名読み	歌 手 名	歌手名読み
CD	9001	ガラスの十代	がらすの	光ＧＥＮＪＩ	ひかるげ
CD	9001	パラダイス銀河	ぱらだい	光ＧＥＮＪＩ	ひかるげ
TP	9002	あいつがとまらない	あいつが	吉田　栄作	よしだえ
TP	9002	夢色のメッセージ	ゆめいろ	西村　知美	にしむら
TP	9002	君は流れ星	きみはな	西村　知美	にしむら
CD	9003	エンゼルフィッシュ	えんぜる	戸川　京子	とがわき
CD	9003	ある微笑	あるびし	戸川　京子	とがわき
CD	9004	いとしのエリー	いとしの	ベンチャーズ	べんちゃ
CD	9005	いつかきっと	いつかき	小泉　今日子	こいずみ
CD	9005	ＬａＬａＬａ	ららら	小泉　今日子	こいずみ
TP	9006	Ｃｈｏｒｕｓ　Ｃｉｔｙ	こーらす	佐藤　宏	さとうひ

TP	9006	明日への啓示	あすへの	エンヤ	えんや
TP	9006	星に願いを	ほしにね	オリビア・ニュート	おりびあ
CD	9007	ＪＵＳＴＩＣＥ	じゃすて	徳永　英明	とくなが
CD	9008	Ｈｅａｖｅｎ　Ｋｎｏｗ	へぶんの	浜田　麻里	はまだ
CD	9008	Ｂｅ　Ｗｉｌｄ	びーわい	浜田　麻里	はまだ
CD	9009	風のロンリーウエイ	かぜのろ	杉山　清貴	すぎやま
CD	9009	サンセット・ラブソング	さんせっ	杉山　清貴	すぎやま
CD	9010	天の子守歌	てんのこ	オユンナ	おゆんな

音楽ソフト管理システムの概要は、次のとおりです。

- 音楽ソフト管理
  - データの入力（新規入力、追加入力）
  - データの一覧表示
  - データの検索、訂正、削除
  - 索引簿（インデックスファイル）の作成
  - 曲名及び歌手名のあいうえお順の画面表示

はじめに、データ入力のプログラムを作ります。

入力に際して、同じデータを繰り返して幾度も入力する場合があるので、その場合は ↵ を押せば直前のデータと同じ内容がコピーされるように改善しました。

また、入力部分の操作を統一して、その部分をサブプロシージャへ移しました。

プログラムの構造は次のとおりです。

- データの入力
  - 準備
    - インクルードファイルの設定
    - 音楽のレコードの型の設定
    - レコード名とその型の設定
    - 音楽ファイルを開く
    - 音楽ファイルのレコード数の計算
  - 繰り返す
    - 媒体の入力
    - 登録番号の入力
    - 曲名の入力
    - 曲名の読みの入力
    - 歌手名の入力
    - 歌手名の読みの入力
      - ［入力のサブプロシージャ］
      - 渡されたデータの保存
      - 漢字を使用するならば、漢字モードＯＮ
      - データの入力
      - 漢字を使用したならば、漢字モードＯＦＦ
      - 必要とするデータの桁数分を切り出す
      - 何も入力されなければ
        - 保存していたデータを使用する
        - 入力時のデータの画面表示を消去して
          - 新しいデータを表示する
        - 入力したデータがEならば
          - 音楽ファイルを閉じて終了する

［メインプロシージャ］

```
DEFINT A-Z
DECLARE SUB NYUURYOKU (X, Y, ANNAI$, ONGAKUDATA$, L, KANZI)
'音楽ソフト管理１　　　　****データの入力****
'$INCLUDE: 'QB.BI'                                'インクルードファイルの設定
'$INCLUDE: 'GEN.BI'
TYPE ONGAKUTYPE                                '音楽ソフトファイルの形式を決める
    BAITAI       AS STRING * 2  '音楽ソフトの媒体（ｺﾝﾊﾟｸﾄﾃﾞｨｽｸ:CD ﾃｰﾌﾟ:TP）
    BANGOU       AS STRING * 4                  '番号を４桁
    KYOKU        AS STRING * 28                 '曲名を全角14桁
    KYOKUYOMI    AS STRING * 8                  '曲名のヨミを全角4桁
    KASYU        AS STRING * 18                 '歌手名を全角9桁
    KASYUYOMI    AS STRING * 8                  '歌手名のヨミを全角4桁
    S            AS INTEGER                     'データの削除のシルシ
END TYPE
DIM ONGAKU AS ONGAKUTYPE                 '音楽ソフトのレコードの容器をつくる
OPEN "B:ONGAKU.DAT" FOR RANDOM AS #1 LEN = LEN(ONGAKU)
                                                '音楽ソフトファイルを開く
I = LOF(1) ¥ LEN(ONGAKU)                        'ファイルのレコード数の計算
ONGAKU.S = -1                                   '削除していないシルシ
CLS                                             '画面消去
PRINT "****音楽ソフトファイルの入力****"
DO
    NYUURYOKU 10, 5,"媒体（ｺﾝﾊﾟｸﾄﾃﾞｨｽｸ:CD ﾃｰﾌﾟ:TP 同じ:ﾘﾀｰﾝ 終了:E） -->",
                                                ONGAKU.BAITAI, 2, 0
    NYUURYOKU 10, 7,"登録番号（同じ:ﾘﾀｰﾝ） -->", ONGAKU.BANGOU, 4, 0
    NYUURYOKU 10, 9,"曲名（同じ:ﾘﾀｰﾝ） -->", ONGAKU.KYOKU, 28, -1
    NYUURYOKU 10,11,"曲名読み（同じ:ﾘﾀｰﾝ） -->", ONGAKU.KYOKUYOMI, 8, -1
    NYUURYOKU 10,13,"歌手名（同じ:ﾘﾀｰﾝ） -->", ONGAKU.KASYU, 18, -1
    NYUURYOKU 10,15,"歌手名読み（同じ:ﾘﾀｰﾝ） -->", ONGAKU.KASYUYOMI, 8, -1
    I = I + 1
    PUT #1, I, ONGAKU
LOOP
```

［データ入力のサブプロシージャ］

```
SUB NYUURYOKU (X, Y, ANNAI$, ONGAKUDATA$, L, KANZI)
                                               'データ入力のサブプロシージャ
HOZON$ = ONGAKUDATA$                            '渡されたデータを保存
IF KANZI THEN                                   '漢字を使用するならば
    Fep "ON"                                    '漢字入力モードＯＮ
END IF
LOCATE Y, X: PRINT ANNAI$; : INPUT "", ONGAKUDATA$          'データ入力する
IF KANZI THEN                                   '漢字を使用したら
    Fep "OFF"                                   '漢字入力モードをＯＦＦ
END IF
```

```
ONGAKUDATA$ = MID$(ONGAKUDATA$, 1, L)
                                      '入力したデータから必要な桁数を切り出す
IF ONGAKUDATA$ = "" THEN                        '何も入力されない場合は
    ONGAKUDATA$ = HOZON$                        '保存していたデータを使う
END IF                                          '入力時の表示を消して
LOCATE Y, X + LEN(ANNAI$):PRINT "                                  "
LOCATE Y, X + LEN(ANNAI$):PRINT ONGAKUDATA$     '切り出したデータを表示する
IF ONGAKUDATA$ = "E" THEN                       '入力データがEならば
    CLOSE #1: END                               'ファイルを閉じて終了する
END IF
END SUB
```

画面表示の例

```
****音楽ソフトファイルの入力****

    媒体（ｺﾝﾊﾟｸﾄﾃﾞｨｽｸ:CD ﾃｰﾌﾟ:TP 同じ:ﾘﾀｰﾝ 終了:E）-->CD
    登録番号（同じ:ﾘﾀｰﾝ）-->9001
    曲名（同じ:ﾘﾀｰﾝ）-->ガラスの十代
    曲名読み（同じ:ﾘﾀｰﾝ）-->がらすの
    歌手名（同じ:ﾘﾀｰﾝ）-->光ＧＥＮＪＩ
    歌手名読み（同じ:ﾘﾀｰﾝ）-->ひかるげ
```

DEFINT A-Z として、一応すべてのアルファベットを整数型の変数とし、実数や文字列の変数の場合は！や$を付して使うことにしました。

I = LOF(1) ¥ LEN(ONGAKU) は、ファイルのレコード数を計算します。LOF(1)は#1のファイルの大きさ（バイト数）を、LEN(ONGAKU)は１レコード（１件分のデータ）の大きさ（バイト数）を計算し、¥は整数の割算をします。

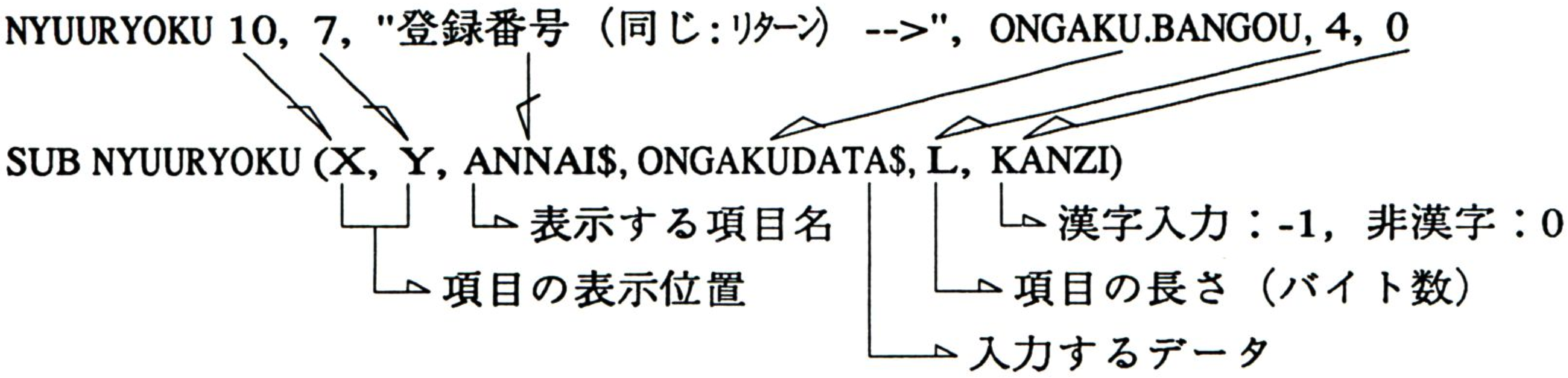
新規のファイルの場合は当然ゼロになるので、このプログラムだけでデータの新規入力にも追加入力にも両方に使用できます。

ただし入力するたびに自動的にデータ追加されますので、ほんとうに新規入力するときは、ＭＳ－ＤＯＳのDELコマンドを使ってファイルを削除してから、このプログラムを実行してください。

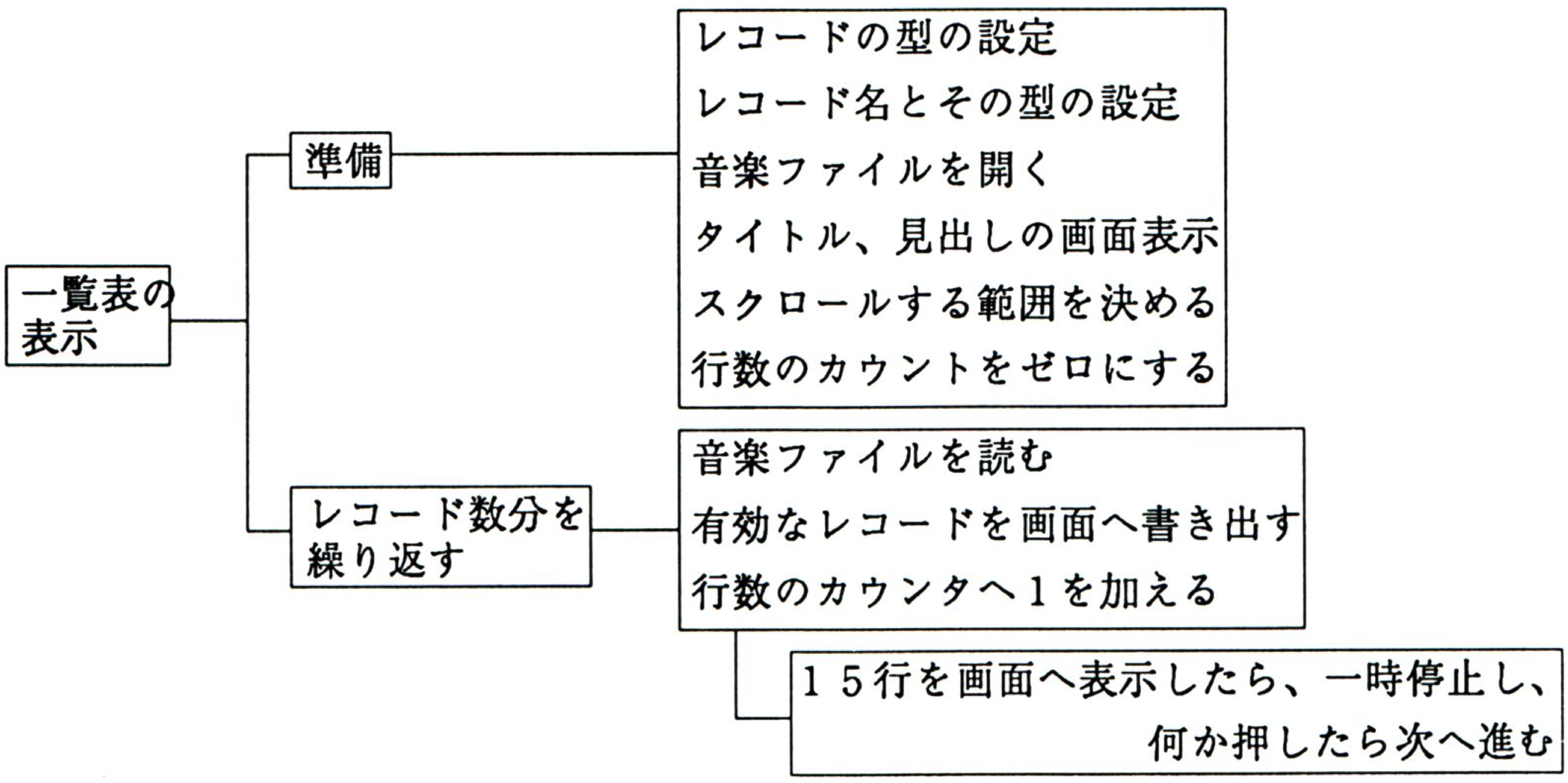
```
A>DEL B:ONGAKU.DAT
```

レコードの削除は、実際には削除しないでONGAKU.S = 0 とすることによって、削除と扱うことにします。ONGAKU.S = -1にしておくことによって、IF ONGAKU.S THEN .... と書いて、もしもONGAKU.Sが真ならば .... とすることができます。

サブプロシージャへは次のようにデータが渡されます。



次、音楽ファイルの一覧表を画面表示するプログラムをつくります。
プログラムの構造は次のとおりです。




```
DEFINT A-Z
'音楽ソフト管理２          ****一覧表の表示****
TYPE ONGAKUTYPE                              '音楽ソフトファイルの形式を決める
     BAITAI      AS STRING * 2   '音楽ソフトの媒体（ｺﾝﾊﾟｸﾄﾃﾞｨｽｸ:CD ﾃｰﾌﾟ:TP）
     BANGOU      AS STRING * 4               '番号を４桁
     KYOKU       AS STRING * 28              '曲名を全角14桁
     KYOKUYOMI   AS STRING * 8               '曲名のヨミを全角4桁
     KASYU       AS STRING * 18              '歌手名を全角9桁
     KASYUYOMI   AS STRING * 8               '歌手名のヨミを全角4桁
     S           AS INTEGER                  'データの削除のシルシ
END TYPE
DIM ONGAKU AS ONGAKUTYPE                     '音楽ソフトのレコードの容器を作る
OPEN "B:ONGAKU.DAT" FOR RANDOM AS #1 LEN = LEN(ONGAKU)
                                             '音楽ソフトファイルを開く
CLS                                          '←画面消去
PRINT "****音楽ソフトファイルの一覧表****"
PRINT
PRINT "番号 媒体 登録番号     曲     名            ";
```

```
PRINT "曲名読み　　歌　手　名　　　歌手名読み"
VIEW PRINT 5 TO 24
N = 0                                      '表示する行数のカウントをゼロ
FOR I = 1 TO LOF(1) ¥ LEN(ONGAKU)          'レコードの数だけ繰り返す
    GET #1, I, ONGAKU                      'ファイルを読む
    IF ONGAKU.S THEN                       '←レコードが有効ならば
        PRINT I; " "; ONGAKU.BAITAI; " "; ONGAKU.BANGOU;
        PRINT " "; ONGAKU.KYOKU; " ";
        PRINT ONGAKU.KYOKUYOMI; " "; ONGAKU.KASYU;
        PRINT " "; ONGAKU.KASYUYOMI
        N = N + 1                          '画面に表示した行数を数える
        IF N MOD 15 = 0 THEN               '１５行を表示したら止める
            PRINT : PRINT "何かキーを押してください。"
            DO
            LOOP UNTIL INKEY$ <> ""        '何かキーを押したら次へ
            CLS
        END IF
    END IF
NEXT I
CLOSE #1                                   'ファイルを閉じる
```

```
VIEW PRINT 5 TO 24
```

画面の５行目から２４行目までの範囲内に表示することです。

画面消去も文字の表示もこの範囲内で行われるので、タイトルと見出しは常にそのまま表示されています。

```
IF ONGAKU.S THEN
```

レコードが有効なものか、削除されたものかを判定しています。

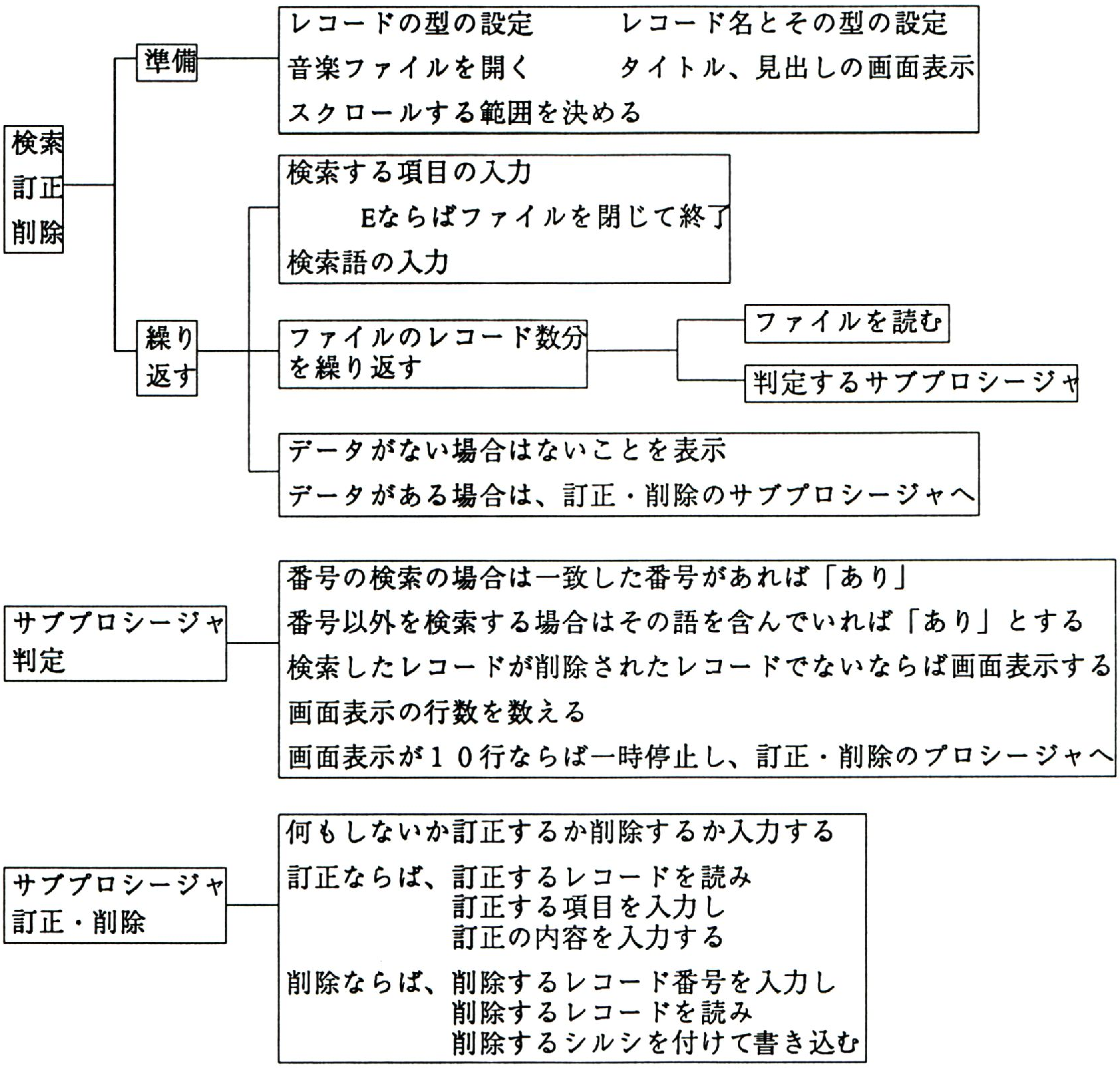
有効なレコードならば画面表示されます。

6．2　検索・訂正・削除をまとめる

次に、検索・訂正・削除のプログラムを作ります。

訂正・削除は、そのレコードを検索してから行いますので、検索のプログラムと訂正・削除のプログラムを組み合せてまとめて作ることができます。

プログラムは次のとおりです。




```
DECLARE SUB TEISEI ()
DEFINT A-Z
DECLARE SUB HANTEI (KENSAKUGO$, ONGAKUDATA$, K$, N, I)
'音楽ソフト管理３          ****データの検索****
'$INCLUDE: 'QB.BI'
'$INCLUDE: 'GEN.BI'
TYPE ONGAKUTYPE                              '音楽ソフトファイルの形式を決める
```

```
    BAITAI       AS STRING * 2    '音楽ソフトの媒体（コンパクトディスク:CD テープ:TP）
    BANGOU       AS STRING * 4                        '番号を４桁
    KYOKU        AS STRING * 28                       '曲名を全角15桁
    KYOKUYOMI    AS STRING * 8                        '曲名のヨミを全角4桁
    KASYU        AS STRING * 18                       '歌手名を全角10桁
    KASYUYOMI    AS STRING * 8                        '歌手名のヨミを全角4桁
    S            AS INTEGER                           'データの削除のシルシ
END TYPE
DIM ONGAKU AS ONGAKUTYPE                   '音楽ソフトのレコードの容器を作る
OPEN "B:ONGAKU.DAT" FOR RANDOM AS #1  LEN = LEN(ONGAKU)
                                                      '音楽ソフトファイルを開く
CLS                                                   '画面消去
PRINT "****音楽ソフトファイルの検索****"
PRINT
PRINT "媒体 番号        曲        名                  ";
PRINT "曲名読み     歌  手  名        歌手名読み"
VIEW PRINT 5  TO 24
DO
INPUT" 検索する項目を指定してください。番号:1 曲名:2 歌手名:3 終了:E ：",
                                             K$        '検索する項目を入力する
    PRINT
    IF K$ = "E" THEN                                  'Eならば
        CLOSE #1:  END                                'ファイルを閉じて終了
    END IF
    IF K$ > "1" THEN                                  '番号以外ならば
        Fep "ON"                                      '漢字入力モードをＯＮ
    END IF
    INPUT "  検索語を入力してください。", KENSAKUGO$      '検索する語の入力
    PRINT
    IF K$ > "1" THEN                                  '漢字入力モードをＯＦＦ
        Fep "OFF"
    END IF
    N = 0
    FOR I  = 1  TO LOF(1) ¥ LEN(ONGAKU)               'レコード数分繰り返す
        GET #1, I, ONGAKU                             '読む
        SELECT CASE K$                               '検索している語があるか判定
            CASE "1":  HANTEI KENSAKUGO$, ONGAKU.BANGOU, K$, N, I
            CASE "2":  HANTEI KENSAKUGO$, ONGAKU.KYOKU, K$, N, I
            CASE "3":  HANTEI KENSAKUGO$, ONGAKU.KASYU, K$, N, I
        END SELECT
    NEXT I
    PRINT
    IF N = 0 THEN
        PRINT "  検索したデータは存在しません。": PRINT   'データがない表示
    ELSE
        TEISEI                              '訂正・削除のサブプロシージャへ
    END IF
LOOP
```

[検索語があるかどうか判定するサブプロシージャ]

```
SUB HANTEI (KENSAKUGO$, ONGAKUDATA$, K$, N, I)      '検索語があるか判定
SHARED ONGAKU AS ONGAKUTYPE                         'メインと共通領域
SW = 0                                              '検索語がないシルシ
SELECT CASE K$
     CASE "1"                                       '番号の検索の場合は
          IF KENSAKUGO$ = ONGAKUDATA$ THEN          '一致すれば検索語あり
               SW = 1                               'あるシルシ
          END IF
     CASE ELSE                                      '番号以外の検索の場合は
          IF KINSTR(ONGAKUDATA$, KENSAKUGO$) > 0 THEN  '検索語を含んでいれば
               SW = 1                               'あるシルシ
          END IF
END SELECT
IF SW = 1 AND ONGAKU.S THEN        '検索語があってレコードが有効ならば画面表示
     PRINT I; " "; ONGAKU.BAITAI; " "; ONGAKU.BANGOU; " ";
     PRINT ONGAKU.KYOKU; " ";
     PRINT ONGAKU.KYOKUYOMI; " "; ONGAKU.KASYU; " ";
     PRINT ONGAKU.KASYUYOMI
     N = N + 1
     IF N MOD 10 = 0 THEN                           '１０行を表示したら止める
          TEISEI                               '訂正・削除のサブプロシージャへ
     END IF
END IF
END SUB
```

[訂正と削除のサブプロシージャ]

```
SUB TEISEI                                          '訂正と削除のサブプロシージャ
'******訂正・削除******
SHARED ONGAKU AS ONGAKUTYPE                         'メインとの共通領域
DO
     PRINT
     PRINT "次のデータ:ｽﾍﾟｰｽ 訂正:C 削除:D -->"
     DO
          J$ = INKEY$                               '訂正と削除の選択
     LOOP UNTIL J$ = " " OR J$ = "C" OR J$ = "D"
     SELECT CASE J$
          CASE " "                                  'スペースならば
               EXIT SUB                             'サブを抜けて戻る
          CASE "C"                                  'Cならば
               INPUT "訂正するレコード番号 -->"; J
                                               '訂正するレコード番号の入力
               GET #1, J, ONGAKU
               INPUT
```

```
"訂正する項目　登録番号:1　曲名:2　曲名読み:3　歌手名:4　歌手名読み:5 -->"
                                              ,T          '訂正する項目の入力
            IF T > 1 THEN
                Fep "ON"                                  '漢字モードＯＮ
            END IF
            SELECT CASE T                                 '訂正項目を入力
                CASE 1
                    PRINT ONGAKU.BANGOU;
                    INPUT "-->", ONGAKU.BANGOU
                CASE 2
                    PRINT ONGAKU.KYOKU;
                    INPUT "-->", ONGAKU.KYOKU
                CASE 3
                    PRINT ONGAKU.KYOKUYOMI;
                    INPUT "-->", ONGAKU.KYOKUYOMI
                CASE 4
                    PRINT ONGAKU.KASYU;
                    INPUT "-->", ONGAKU.KASYU
                CASE 5
                    PRINT ONGAKU.KASYUYOMI;
                    INPUT "-->", ONGAKU.KASYUYOMI
            END SELECT
            IF T > 1 THEN
                Fep "OFF"                                 '漢字モードＯＦＦ
            END IF
            PUT #1, J, ONGAKU                         '訂正済みのレコードを書き込む
        CASE "D"
            INPUT "削除するレコード番号 -->"; J
                                                        '削除するレコード番号を入力
            GET #1, J, ONGAKU                             '削除するレコードを読む
            ONGAKU.S = 0                                  '削除するシルシをつける
            PUT #1, J, ONGAKU                           '削除するレコードを書き込む
    END SELECT
LOOP
END SUB
```

検索語（検索に使うデータ）を入力する前に、それが属する項目を番号で指定しました。この番号によって、漢字入力モードのＯＮ、ＯＦＦをするようにします。

検索語があるかどうかの判定部分はサブプロシージャにしましたから、番号、曲名、歌手名の検索はこのサブプロシージャを共同利用することによって行います。

判定のサブプロシージャには、次のようにデータを受け渡します。

HANTEI KENSAKUGO$, ONGAKU.KYOKU, K$, N, I

SUB HANTEI (KENSAKUGO$, ONGAKUDATA$, K$, N, I)

- I → レコード番号
- N → 検索しているデータの有無（あり:>=1　なし:0）
- K$ → 入力した検索語の項目番号
- KENSAKUGO$ → 入力した検索語
- ONGAKUDATA$ → ファイルから読んだ検索対象項目のデータ

判定のサブプロシージャでは、メインから受け取ったデータに従って、検索語がファイルから読んだレコードに存在するかどうかを判断します。番号の場合は完全一致のみ、他のデータの場合は検索語を含んでいればよいということにしました。検索したレコードは画面表示します。該当するレコードが多いと、画面がスクロールして画面から消えてしまうので、１０行を表示したところで一時停止して、次のデータを見るか、訂正・削除をするかを入力します。もしも訂正・削除をするならば、訂正・削除のサブプロシージャを使います。

メインプロシージャは、判定のサブプロシージャから渡されたデータによって、検索したレコードが存在すれば訂正・削除のサブプロシージャを使い、存在しなければそのことを表示します。

訂正・削除のサブプロシージャは、まず、次の検索か、訂正か、削除かを選択します。

次の検索ならサブプロシージャを抜け出し、訂正なら項目ごとに新たなデータを入力して書き戻し、削除なら削除のシルシを付けて書き戻します。

訂正・削除のサブプロシージャのデータは、SHARED ONGAKU AS ONGAKUTYPE として、メインプロシージャのデータを共同利用しています。

６．３　インデックスファイル

今度は、音楽ファイルを、曲名、歌手名の「あいうえお順」で表示するプログラムを作ります。もちろん、音楽ファイルそのものを曲名順あるいは歌手名順に並べなおして画面表示することもできますが、ここでは、インデックスファイル（索引簿）を作ってこれを参照しながら画面表示する方法について、学ぶことにします。

インデックスファイルは、レコードを読む（表示する）順番をメモしてあるファイルです。例えば草野球の打順を思い浮かべてください。各選手が全員の打順を知らなくても、それぞれ自分が「誰の次」かを分っていればプレーができます。このように「次は誰」かだけを記録したファイルを用意して、これをたどっていくことによって、必要な順序でファイルを読み取ります。

インデックスファイルについて理解しやすいように、４個の身長のデータを使って、低い身長から高い身長へ順にデータを取り出すことができるインデックスファイルを作る手順を説明します。

生徒番号	身長（ｃｍ）
1	175
2	172
3	181
4	162

1. インデックスファイルを表の形で用意します。

次の身長の容器番号の容器（＝索引番号）	SAKUIN(0)	SAKUIN(1)	SAKUIN(2)	SAKUIN(3)	SAKUIN(4)
身長データの容器	SINTYOU(0)	SINTYOU(1)	SINTYOU(2)	SINTYOU(3)	SINTYOU(4)

2. 生徒番号1の身長をSINTYOU(1)に、2の身長をSINTYOU(2)に、というように、身長データの容器に数値を入れておきます。

次の身長の容器番号の容器（＝索引番号）	SAKUIN(0)	SAKUIN(1)	SAKUIN(2)	SAKUIN(3)	SAKUIN(4)
身長データの容器	SINTYOU(0)	SINTYOU(1) 175	SINTYOU(2) 172	SINTYOU(3) 181	SINTYOU(4) 162

3. いよいよ、索引番号を付けます。最初は容器1までです。

SAKUIN(0)はスタートの容器番号ですが、1までの索引ですからここでは1です。

SAKUIN(1)には、この位置に対応する容器SINTYOU(1)が最後のデータ（現時点で一番大きいデータ）であるシルシとして、0を入れます。

次の身長の容器番号の容器（=索引番号）	SAKUIN(0) 1	SAKUIN(1) 0	SAKUIN(2)	SAKUIN(3)	SAKUIN(4)
身長データの容器	SINTYOU(0)	SINTYOU(1) 175	SINTYOU(2) 172	SINTYOU(3) 181	SINTYOU(4) 162

4. これからは、データを比較しながら索引番号を付けます。まず容器2までです。

スタートの容器番号を示すSAKUIN(0)が1なので、SINTYOU(1)[=175]と新しく比較するSINTYOU(2)[=172]とを比較します。

SINTYOU(2)のほうが小さいので、SAKUIN(2)へ1を入れて、SAKUIN(0)を1から2に替えます。

次の身長の容器番号の容器（=索引番号）	SAKUIN(0) 2	SAKUIN(1) 0	SAKUIN(2) 1	SAKUIN(3)	SAKUIN(4)
身長データの容器	SINTYOU(0) スタート	SINTYOU(1) 175	SINTYOU(2) 172	SINTYOU(3) 181	SINTYOU(4) 162

5. 容器3までの索引番号です。

SAKUIN(0)が2なので、SINTYOU(2)[=172]と、新しく比較するSINTYOU(3)[=181]とを比較します。

SINTYOU(2)[=172]のほうが小さいので、SAKUIN(2)を見ます。

SAKUIN(2)は1なので、SINTYOU(1)[=175]とSINTYOU(3)[=181]とを比較します。

SINTYOU(1)[=175]のほうが小さいので、SAKUIN(1)を見ます。

SAKUIN(1)は、最後のデータ（現時点で最大のデータ）のシルシである0なので、SAKUIN(3)へ0を入れて、SAKUIN(1)を0から3に替えます。

次の身長の容器番号の容器（=索引番号）	SAKUIN(0) 2	SAKUIN(1) 3	SAKUIN(2) 1	SAKUIN(3) 0	SAKUIN(4)
身長データの容器	SINTYOU(0) スタート	SINTYOU(1) 175	SINTYOU(2) 172	SINTYOU(3) 181	SINTYOU(4) 162

6. 容器4までの索引番号です。これですべての索引番号が付きます。

SAKUIN(0)が2なので、SINTYOU(2)[=172]と、新しく比較するSINTYOU(4)[=162]とを比較します。

SINTYOU(4)のほうが小さいので、SAKUIN(4)へ2を入れて、SAKUIN(0)を2から4に替えます。

次の身長の容器番号の容器（＝索引番号）	SAKUIN(0) 4	SAKUIN(1) 3	SAKUIN(2) 1	SAKUIN(3) 0	SAKUIN(4) 2
身長データの容器	SINTYOU(0) スタート	SINTYOU(1) 175	SINTYOU(2) 172	SINTYOU(3) 181	SINTYOU(4) 162

これで完成です。どうぞたどってみてください。

1． スタート位置のSAKUIN(0)は4なので、SINTYOU(4)[=162]を読みます。

2． SAKUIN(4)は2なので、SINTYOU(2)[=172]を読みます。

3． SAKUIN(2)は1なので、SINTYOU(1)[=175]を読みます。

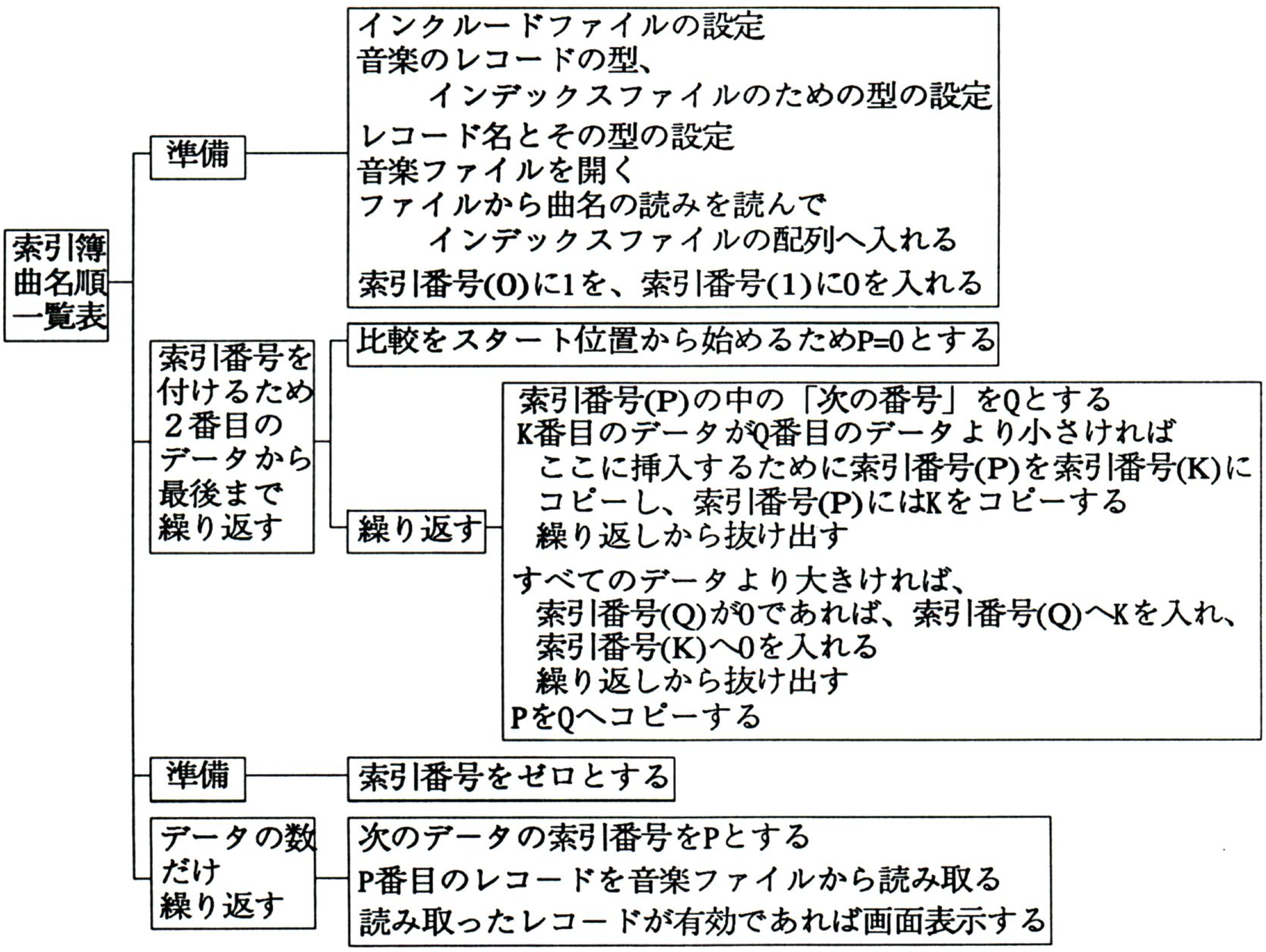
4． SAKUIN(1)は3なので、SINTYOU(3)[=181]を読みます。

5． SAKUIN(3)は0なので、これで終りということです。

どうですか、低い順に読むことができたでしょう。

このように、インデックスファイルは索引番号を書きなおすことによって作ることができます。これなら、データの順序を何度も書きなおす「分類＝ソート」という処理を行わないで同じようなことができるので、処理時間は大きく短縮できます。

それでは、曲名の読みの「あいうえお順」の索引簿を作って一覧表を画面表示するプログラムを作ります。プログラムの構造は、次のとおりです。



```
DEFINT A-Z
'音楽ソフト管理４         ****曲名索引簿・曲名順一覧表****
'$INCLUDE: 'QB.BI'
'$INCLUDE: 'GEN.BI'
TYPE ONGAKUTYPE                               '管理ファイルの形式を決める
    BAITAI       AS STRING * 2                '媒体
    BANGOU       AS STRING * 4                '番号を４桁とする
    KYOKU        AS STRING * 28               '曲名を全角15桁とする
    KYOKUYOMI    AS STRING * 8                '曲名のヨミを全角４桁とする
    KASYU        AS STRING * 18               '歌手名を全角10桁とする
    KASYUYOMI    AS STRING * 8                '歌手名のヨミを全角10桁とする
    S            AS INTEGER                   'データの生死のシルシ
END TYPE
TYPE SAKUINTYPE                               '索引簿の形式を決める
    BANGOU       AS INTEGER                   '次データの位置を整数とする
    KYOKUYOMI    AS STRING * 8                '曲名の読みを全角４桁とする
END TYPE
DIM ONGAKU AS ONGAKUTYPE                      '管理ファイルの容器を作る
DIM SAKUIN(0 TO 1000) AS SAKUINTYPE           '索引簿は0から1000までの容器
OPEN "B:ONGAKU.DAT" FOR RANDOM AS #1 LEN = LEN(ONGAKU)
                                              '管理ファイルを開く
CLS                                           '画面消去
PRINT "****曲名索引簿作成****"
PRINT
FOR I = 1 TO LOF(1) ¥ LEN(ONGAKU)
    GET #1, I, ONGAKU                     'ファイルから曲名の読みを読み取る
    SAKUIN(I).KYOKUYOMI = ONGAKU.KYOKUYOMI '曲名の読みを索引簿へ入れる
NEXT I
'*******索引番号を付ける
SAKUIN(0).BANGOU = 1                          '索引簿の0番目の番号を1とする
                                  '↑次のデータが1番目にあることを示す
SAKUIN(1).BANGOU = 0                          '索引簿の1番目の番号を0とする
                          '↑索引番号のついた最後のデータであることを示す
FOR K = 2 TO LOF(1) ¥ LEN(ONGAKU)
                              '２番目のデータからデータの数だけ繰り返す
    p = 0                                     'Pを0にしておく
    DO                                        '繰り返す
        Q = SAKUIN(p).BANGOU                  'Qに索引簿P番目の番号を入れる
            '↓P番目とQ番目のデータの間にK番目のデータが挿入されるとき
        IF SAKUIN(K).KYOKUYOMI < SAKUIN(Q).KYOKUYOMI THEN
                      'K番目の曲名の読みがQ番目の曲名の読みより小さければ
            SAKUIN(K).BANGOU = SAKUIN(p).BANGOU
                                              'P番目の番号をK番目に入れて
            SAKUIN(p).BANGOU = K              'P番目の番号へKを入れて
            EXIT DO                           '繰り返しを中止する
        END IF
            '↓K番目のデータが番号を付けたすべてのデータより大きければ
        IF SAKUIN(Q).BANGOU = 0 THEN          'Q番目の番号が0ならば
```

```
                SAKUIN(Q).BANGOU = K              'Q番目の番号へKを入れて
                SAKUIN(K).BANGOU = 0              'K番目の番号を0にして
                EXIT DO                           '繰り返しを中止する
          END IF
          p = Q                                   'PにQを入れる
     LOOP
NEXT K
PRINT "        ***** 音楽ソフト  曲名順一覧表 *****"
PRINT
P = 0
N = 0
FOR I = 1 TO LOF(1) ¥ LEN(ONGAKU)
     p = SAKUIN(p).BANGOU                         '次のデータの索引番号をPとする
     GET #1, p, ONGAKU                            'P番目のデータを読む
     IF ONGAKU.S THEN                             'もしも生きているデータならば
          PRINT I;"  ";ONGAKU.BAITAI;"  ";ONGAKU.BANGOU;"  ";ONGAKU.KYOKU;
          PRINT ONGAKU.KYOKUYOMI;"  ";ONGAKU.KASYU;ONGAKU.KASYUYOMI
          N = N + 1
     END IF                                       'データを表示する
NEXT I
```

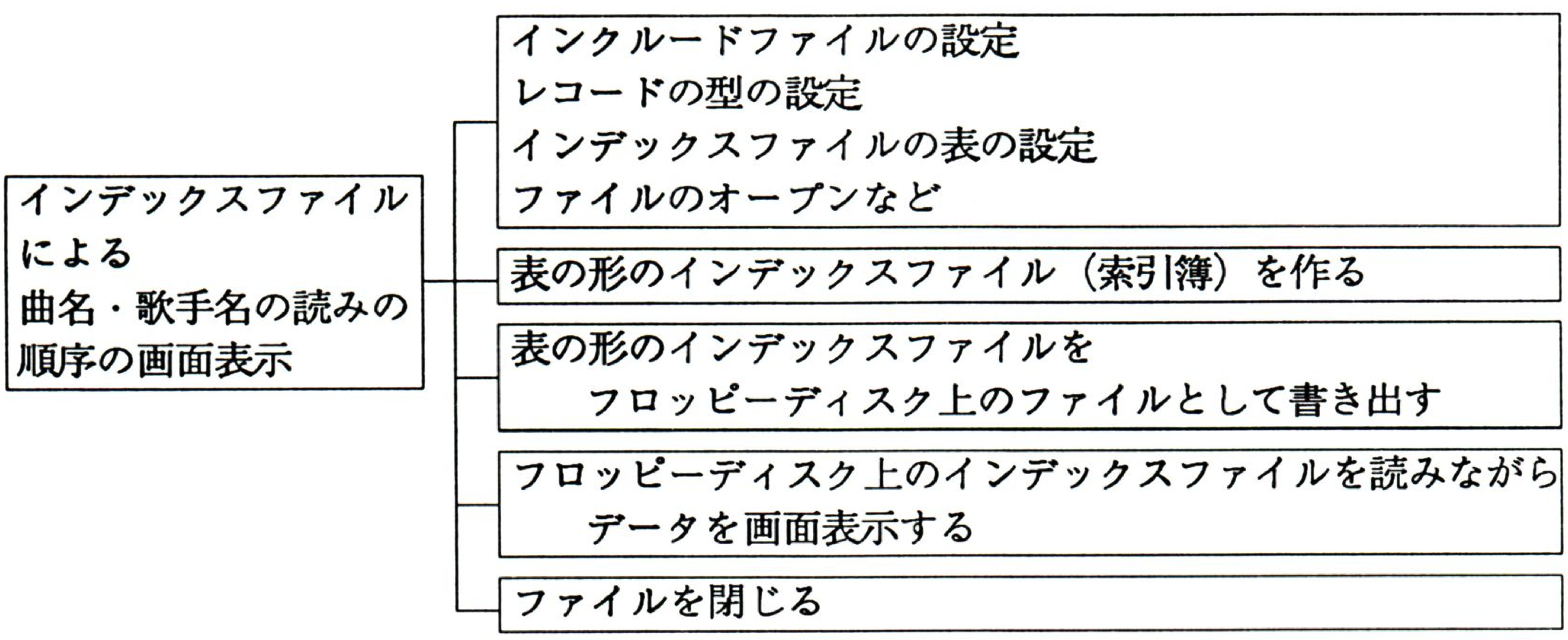
音楽ソフト管理４のプログラムは、インデックスファイルを作る基本形を使って、曲名の読みの順序で画面表示しました。

次は歌手名の読みの順序で画面表示するプログラムを作りますが、ここでは次の事項についてプログラムを改造します。

1. できるだけサブプロシージャへ書き換えて、曲名の読みと歌手名の読みの仕事を共通のプログラムで処理できるようにします。このサブプロシージャは、他のプログラムを作る場合にも利用できるようにします。

2. メモリー内の表の形のインデックスファイルを、フロッピーディスクへファイルとして書き出して、新しいデータが加えられない限り、フロッピーディスク上のインデックスファイルを利用できるようにします。

以上の改造をしたプログラムの構造の概要は、次のとおりです。



```
DECLARE SUB INDKAKU (SAKUIN() AS ANY, FILE$)
DECLARE SUB INDYOMIITIRAN (SAKUIN() AS ANY, FILE$, TITLE$)
DECLARE SUB SAKUINTUKURI (SAKUIN() AS ANY)
DEFINT A-Z
'音楽ソフト管理５      ****索引簿(曲名・歌手名) ファイルの作成****
'$INCLUDE: 'QB.BI'
'$INCLUDE: 'GEN.BI'
TYPE ONGAKUTYPE                               '管理ファイルの形式を決める
    BAITAI       AS STRING * 2                '媒体
    BANGOU       AS STRING * 4                '番号を４桁とする
    KYOKU        AS STRING * 28               '曲名を全角15桁とする
    KYOKUYOMI    AS STRING * 8                '曲名のヨミを全角４桁とする
    KASYU        AS STRING * 18               '歌手名を全角10桁とする
    KASYUYOMI    AS STRING * 8                '歌手名のヨミを全角10桁とする
    S            AS INTEGER                   'データの生死のシルシ
END TYPE
TYPE SAKUINTYPE                               '索引簿の形式を決める
    BANGOU       AS INTEGER                   '次データの位置を整数とする
    ONGAKUDATA   AS STRING * 8                '曲名の読みを全角４桁とする
END TYPE
DIM ONGAKU AS ONGAKUTYPE                      '管理ファイルの容器を作る
DIM KYOKUSAKUIN(0 TO 1000) AS SAKUINTYPE
                                   '索引簿は0から1000までの番号をつけた容器
DIM KASYUSAKUIN(0 TO 1000) AS SAKUINTYPE
                                   '索引簿は0から1000までの番号をつけた容器
CLS                                           '画面表示
PRINT "****索引簿（インデックスファイル）作成****"
PRINT
OPEN "B:ONGAKU.DAT" FOR RANDOM AS #1 LEN = LEN(ONGAKU)
                                              '管理ファイルを開く
'↓ファイルから曲名の読みを読み取る
FOR I = 1 TO LOF(1) ¥ LEN(ONGAKU)
    GET #1, I, ONGAKU                         '管理ファイルからデータを読み取る
    KYOKUSAKUIN(I).ONGAKUDATA = ONGAKU.KYOKUYOMI  'データを索引簿へコピー
    KASYUSAKUIN(I).ONGAKUDATA = ONGAKU.KASYUYOMI
NEXT I
SAKUINTUKURI KYOKUSAKUIN()                    '索引簿を作る
SAKUINTUKURI KASYUSAKUIN()
INDKAKU KYOKUSAKUIN(), "B:KYOKU.IND"
                                    '索引簿ファイルをフロッピーへ書き込む
INDKAKU KASYUSAKUIN(), "B:KASYU.IND"
INDYOMIITIRAN KYOKUSAKUIN(), "B:KYOKU.IND",
                        "       ***** 音楽ソフト  曲名順一覧表 *****"
INDYOMIITIRAN KASYUSAKUIN(), "B:KASYU.IND",
                        "       ***** 音楽ソフト  歌手順一覧表 *****"
CLOSE #1
END
```

［メモリー内の索引簿をフロッピーへ書き出すサブプロシージャ］

```
SUB INDKAKU (SAKUIN() AS SAKUINTYPE, FILE$)
                         'メモリー内の索引簿をフロッピーへ書き出すプロシージャ
OPEN FILE$ FOR RANDOM AS #2 LEN = 10        '索引簿を開く
FOR I = 0 TO LOF(1) ¥ LEN(ONGAKU)           'データの数だけ繰り返す
    PUT #2, I + 1, SAKUIN(I)                '表からフロッピーへ書き出す
NEXT I
CLOSE #2                                    '索引簿を閉じる
END SUB
```

［索引簿をフロッピーから読み取って一覧表を画面表示するサブプロシージャ］

```
SUB INDYOMIITIRAN (SAKUIN() AS SAKUINTYPE, FILE$, TITLE$)
SHARED ONGAKU AS ONGAKUTYPE                 '音楽レコードはメインと共同利用する
CLS                                         '画面消去
PRINT TITLE$                                'タイトル表示
PRINT
OPEN FILE$ FOR RANDOM AS #2 LEN = 10        '索引簿を開く
FOR I = 0 TO LOF(1) ¥ LEN(ONGAKU)           'ゼロからデータの数まで繰り返す
    GET #2, I + 1, SAKUIN(I)                '索引簿を読む
NEXT I
CLOSE #2
P = 0
N = 0
FOR I = 1 TO LOF(1) ¥ LEN(ONGAKU)           'データの数だけ繰り返す
    P = SAKUIN(P).BANGOU                    '次のデータの索引番号をPとする
    GET #1, P, ONGAKU                       'P番目のデータを読む
    IF ONGAKU.S THEN                        'もしも生きているデータならば
        PRINT I;" ";ONGAKU.BAITAI;" ";ONGAKU.BANGOU;" ";ONGAKU.KYOKU;
        PRINT ONGAKU.KYOKUYOMI;" ";ONGAKU.KASYU;ONGAKU.KASYUYOMI
        N = N + 1                           '表示したデータ数を数える
        IF N MOD 10 = 0 THEN                '画面表示を１０行毎に一時停止する
            PRINT "何かキーを押してください。"
            DO
            LOOP UNTIL INKEY$ <> ""         '何か押すと次へ進む
        END IF
    END IF
NEXT I
PRINT "何かキーを押してください。"          '表示が終了したら一時停止
DO
LOOP UNTIL INKEY$ <> ""                     '何か押すと次へ進む
END SUB
```

［索引簿を表の形に作るサブプロシージャ］

```
SUB SAKUINTUKURI (SAKUIN() AS SAKUINTYPE)
SHARED ONGAKU AS ONGAKUTYPE                    '音楽レコードはメインと共同利用
SAKUIN(0).BANGOU = 1                           '索引簿の0番目の番号を1とする
                                               '次のデータが1番目にあることを示す
SAKUIN(1).BANGOU = 0                           '索引簿の1番目の番号を0とする
                          '索引番号のついた最後のデータであることを示す
   FOR K = 2 TO LOF(1) ¥ LEN(ONGAKU)            '２番目からデータの数だけ繰り返す
      P = 0                                    'Pを0にしておく
      DO                                       '繰り返す
         Q = SAKUIN(P).BANGOU
                          '索引番号P番目のデータがもつ索引番号をQとする
         'P番目とQ番目のデータの間にK番目のデータが挿入されるときは
         IF SAKUIN(K).ONGAKUDATA < SAKUIN(Q).ONGAKUDATA THEN
                                     'K番目の読みがQ番目の読みより小さければ
            SAKUIN(K).BANGOU = SAKUIN(P).BANGOU
                                       'P番目の番号をK番目の番号にコピーして
            SAKUIN(P).BANGOU = K       'P番目の番号へはKをコピーして
            EXIT DO                    '繰り返しを中止する
         END IF
         'K番目のデータが番号を付けたすべてのデータよりも大きいときは
         IF SAKUIN(Q).BANGOU = 0 THEN                   'Q番目の番号が0ならば
            SAKUIN(Q).BANGOU = K       'KをQ番目の番号へコピーし
            SAKUIN(K).BANGOU = 0       'K番目の番号を0にし
            EXIT DO                    '繰り返しを中止する
         END IF
         P = Q                         'QをPへコピーする
      LOOP
   NEXT K
END SUB
```

6．4　音楽ソフト管理のまとめ

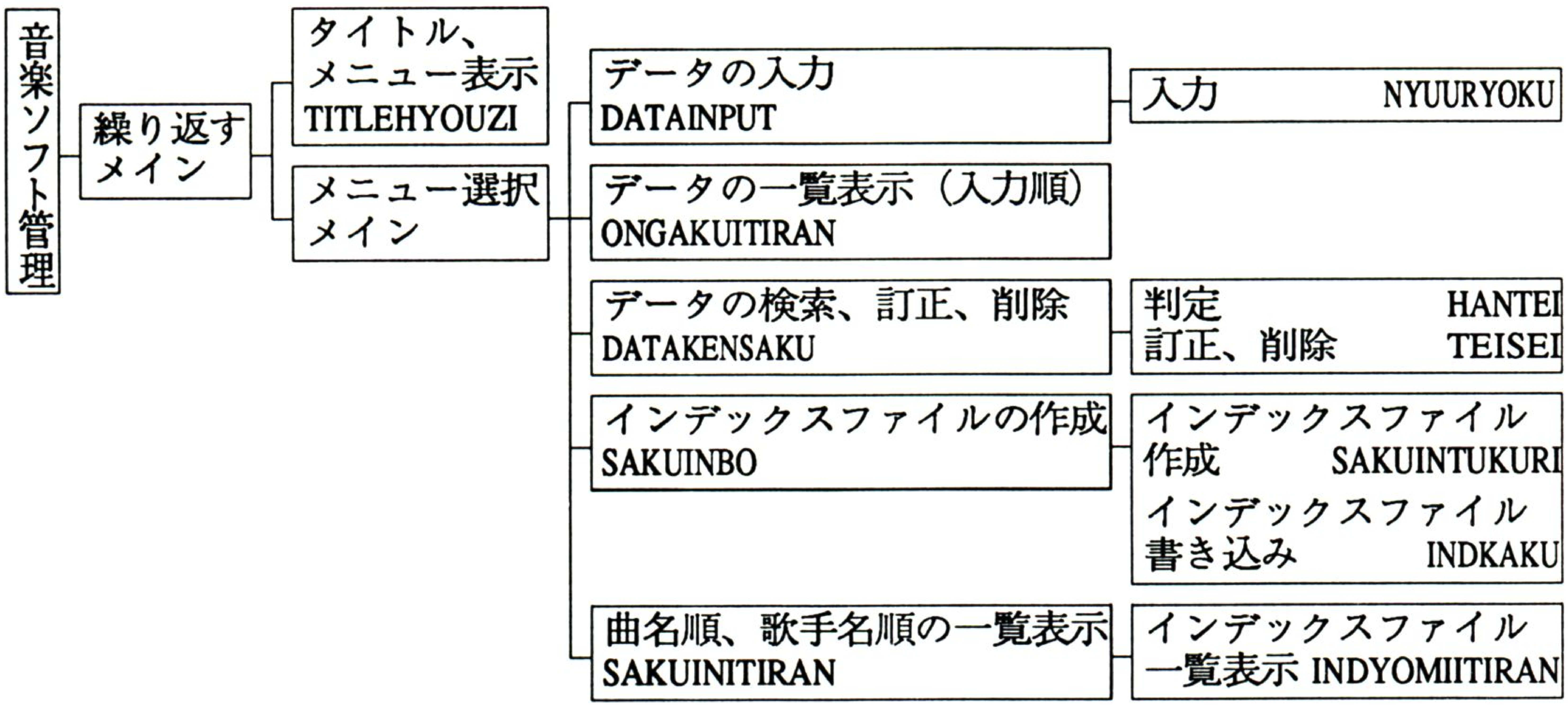
音楽ソフト管理プログラムをまとめます。

これまで作ってきた各プログラムをサブプロシージャに改造し、機能を選択するシンプルなメインプロシージャとタイトルとメニューを表示するサブプロシージャを加えました。

KENSAKUをサブプロシージャに変更すると、HANTEIへはONGAKUレコードをSHAREDでは渡すことができなくなるので、HANTEIの引数を一部変えました。

また、SHARED ONGAKU AS ONGAKUTYPE と DIM ONGAKU AS ONGAKUTYPE のどちらでもよい部分は、後者に統一しました。

プログラムの構造の概要は次のとおりです。




```
DEFINT A-Z
DECLARE SUB SAKUINBO ()
DECLARE SUB SAKUINITIRAN ()
DECLARE SUB TITLEHYOUZI ()
DECLARE SUB TEISEI ()
DECLARE SUB HANTEI (ONGAKU AS ANY,KENSAKUGO$,ONGAKUDATA$,K$,N,I)
DECLARE SUB DATAINPUT ()
DECLARE SUB ONGAKUITIRAN ()
DECLARE SUB DATAKENSAKU ()
DECLARE SUB NYUURYOKU (X,Y,ANNAI$,ONGAKUDATA$,L,KANZI)
DECLARE SUB INDKAKU (SAKUIN() AS ANY,FILE$)
DECLARE SUB INDYOMIITIRAN (SAKUIN() AS ANY,FILE$,TITLE$)
DECLARE SUB SAKUINTUKURI (SAKUIN() AS ANY)
'音楽ソフト管理          メインプロシージャ
'$INCLUDE: 'QB.BI'                    'インクルードファイルの設定
'$INCLUDE: 'GEN.BI'
TYPE ONGAKUTYPE                        '音楽ソフトファイルの形式を決める
     BAITAI        AS STRING * 2  '音楽ソフトの媒体（ｺﾝﾊﾟｸﾄﾃﾞｨｽｸ:CD ﾃｰﾌﾟ:TP）
```

```
    BANGOU      AS STRING * 4             '番号を4桁
    KYOKU       AS STRING * 28            '曲名を全角14桁
    KYOKUYOMI   AS STRING * 8             '曲名のヨミを全角4桁
    KASYU       AS STRING * 18            '歌手名を全角9桁
    KASYUYOMI   AS STRING * 8             '歌手名のヨミを全角4桁
    S           AS INTEGER                'データの削除のシルシ
END TYPE
TYPE SAKUINTYPE                           '索引簿の形式を決める
    BANGOU AS INTEGER                     '次データの位置を整数とする
    ONGAKUDATA AS STRING * 8              '曲名の読みを全角4桁とする
END TYPE
DO                                        '繰り返す
    TITLEHYOUZI                           'タイトルの画面表示
    DO
        K$ = INKEY$                       '番号の入力
        LOOP UNTIL K$ >= "1" AND K$ <= "6"
        SELECT CASE K$
            CASE "1": DATAINPUT           'データの入力
            CASE "2": ONGAKUITIRAN        '一覧表示（入力順）
            CASE "3": DATAKENSAKU         '検索、訂正、削除
            CASE "4": SAKUINBO            'インデックスファイルの作成
            CASE "5": SAKUINITIRAN        '曲名順、歌手名順一覧表示
            CASE "6": END                 '終了
        END SELECT
LOOP
END
```

［データ入力のサブプロシージャ］

```
SUB DATAINPUT                             'データの入力
DIM ONGAKU AS ONGAKUTYPE                  '音楽ソフトのレコードの容器を作る
OPEN "B:ONGAKU.DAT" FOR RANDOM AS #1 LEN = LEN(ONGAKU)
                                          '音楽ソフトファイルを開く
I = LOF(1) ¥ LEN(ONGAKU)                  'ファイルのレコード数の計算
ONGAKU.S = -1                             '削除していないシルシ
CLS                                       '画面消去
PRINT "****音楽ソフトファイルの入力****"
DO
    NYUURYOKU 10,5,"媒体（ｺﾝﾊﾟｸﾄﾃﾞｨｽｸ:CD ﾃｰﾌﾟ:TP 同じ:ﾘﾀｰﾝ 終了:E） -->",
                                          ONGAKU.BAITAI, 2, 0
    IF ONGAKU.BAITAI = "E " THEN          '入力データがEならば
        CLOSE #1: EXIT SUB                'ファイルを閉じて終了する
    END IF
    NYUURYOKU 10, 7, "登録番号（同じ:ﾘﾀｰﾝ） -->", ONGAKU.BANGOU,4,0
    NYUURYOKU 10, 9, "曲名（同じ:ﾘﾀｰﾝ） -->", ONGAKU.KYOKU, 28, -1
    NYUURYOKU 10, 11, "曲名読み（同じ:ﾘﾀｰﾝ） -->",ONGAKU.KYOKUYOMI, 8, -1
    NYUURYOKU 10, 13, "歌手名（同じ:ﾘﾀｰﾝ） -->", ONGAKU.KASYU,18,-1
    NYUURYOKU 10, 15, "歌手名読み（同じ:ﾘﾀｰﾝ） -->",ONGAKU.KASYUYOMI, 8, -1
```

```
    I = I + 1
    PUT #1, I, ONGAKU
LOOP
END SUB
```

［データ検索のサブプロシージャ］

```
SUB DATAKENSAKU                                    '****データの検索****
DIM ONGAKU AS ONGAKUTYPE                           '音楽ソフトのレコードの容器を作る
OPEN "B:ONGAKU.DAT" FOR RANDOM AS #1 LEN = LEN(ONGAKU)
                                                   '音楽ソフトファイルを開く
CLS                                                '画面消去
PRINT "****音楽ソフトファイルの検索****"
PRINT
PRINT "番号 媒体      曲      名              曲名読み    ";
                                                "歌  手  名      歌手名読み"
VIEW PRINT 5 TO 24
DO
INPUT" 検索する項目を指定してください。番号:1 曲名:2 歌手名:3 終了:E : ",
                                  K$               '検索する項目を入力する
    PRINT
    IF K$ = "E" THEN                               'Eならば
        CLOSE #1: VIEW PRINT 1 TO 25: EXIT SUB          'ファイルを閉じて終了
    END IF
    IF K$ > "1" THEN                               '番号以外ならば
        Fep "ON"                                   '漢字入力モードをON
    END IF
    INPUT "  検索語を入力してください。", KENSAKUGO$      '検索する語の入力
    PRINT
    IF K$ > "1" THEN                               '漢字入力モードをOFF
        Fep "OFF"
    END IF
    N = 0
    FOR I = 1 TO LOF(1) ¥ LEN(ONGAKU)              'ファイルのレコード数分繰り返す
        GET #1, I, ONGAKU                          '読む
        SELECT CASE K$                             '検索している語があるか判定
            CASE "1"
                HANTEI ONGAKU, KENSAKUGO$, ONGAKU.BANGOU, K$, N, I
            CASE "2"
                HANTEI ONGAKU, KENSAKUGO$, ONGAKU.KYOKU, K$, N, I
            CASE "3"
                HANTEI ONGAKU, KENSAKUGO$, ONGAKU.KASYU, K$, N, I
        END SELECT
    NEXT I
    PRINT
    IF N = 0 THEN
        PRINT "  検索したデータは存在しません。"           'データがない表示
        PRINT
```

```
    ELSE
        TEISEI
    END IF
LOOP
END SUB
```

［検索語があるか判定するサブプロシージャ］

```
SUB HANTEI (ONGAKU AS ONGAKUTYPE, KENSAKUGO$, ONGAKUDATA$, K$, N, I)
                                                  '検索語があるか判定
SW = 0                                            '検索語がないシルシ
SELECT CASE K$
    CASE "1"                                      '番号の検索の場合は
        IF KENSAKUGO$ = ONGAKUDATA$ THEN          '一致すれば検索語あり
            SW = 1                                'あるシルシ
        END IF
    CASE ELSE                                     '番号以外の検索の場合は
        IF KINSTR(ONGAKUDATA$, KENSAKUGO$) > 0 THEN   '検索語を含んでいれば
            SW = 1                                'あるシルシ
        END IF
END SELECT
IF SW = 1 AND ONGAKU.S THEN          '検索語があってレコードが有効ならば画面表示
    PRINT  I;"   ";ONGAKU. BAITAI;"  ";ONGAKU. BANGOU;"  ";ONGAKU. KYOKU;"  ";
    PRINT  ONGAKU. KYOKUYOMI;"  ";ONGAKU. KASYU;"  ";ONGAKU. KASYUYOMI
    N = N + 1
    IF N MOD 10 = 0 THEN                          '10行を表示したら止める
        TEISEI                                    '訂正・削除のサブプロシージャへ
    END IF
END IF
END SUB
```

［索引簿をフロッピーへ書き出すサブプロシージャ］

```
SUB INDKAKU (SAKUIN() AS SAKUINTYPE, FILE$)    '索引簿をフロッピーへ書き出す
OPEN FILE$ FOR RANDOM AS #2 LEN = 10          '索引簿を開く
FOR I = 0 TO LOF(1) ¥ LEN(ONGAKU)             'データの数だけ繰り返す
    PUT #2, I + 1, SAKUIN(I)                  '表からフロッピーへ書き出す
NEXT I
CLOSE #2                                      '索引簿を閉じる
END SUB
```

［索引簿をフロッピーから読み取って一覧表を画面表示するのサブプロシージャ］

```
SUB INDYOMIITIRAN (SAKUIN() AS SAKUINTYPE, FILE$, TITLE$)
DIM ONGAKU AS ONGAKUTYPE
CLS                                           '画面消去
```

```
PRINT TITLE$                                    'タイトル表示
PRINT
OPEN FILE$ FOR RANDOM AS #2 LEN = 10            '索引簿を開く
FOR I = 0 TO LOF(1) ¥ LEN(ONGAKU)               'ゼロからデータの数まで繰り返す
    GET #2, I + 1, SAKUIN(I)                    '索引簿を読む
NEXT I
CLOSE #2
P = 0
N = 0
FOR I = 1 TO LOF(1) ¥ LEN(ONGAKU)               'データの数だけ繰り返す
    P = SAKUIN(P).BANGOU                        '次のデータの索引番号をPとする
    GET #1, P, ONGAKU                           'P番目のデータを読む
    IF ONGAKU.S THEN                            'もしも生きているデータならば
        PRINT I;"  ";ONGAKU. BAITAI;" ";ONGAKU. BANGOU;" ";ONGAKU. KYOKU;
        PRINT ONGAKU. KYOKUYOMI;" ";ONGAKU. KASYU;" ";ONGAKU. KASYUYOMI
        N = N + 1                               '表示したデータ数を数える
        IF N MOD 10 = 0 THEN                    '画面表示を10行毎に一時停止する
            PRINT "何かキーを押してください。"
            DO
            LOOP UNTIL INKEY$ <> ""             '何か押すと次へ進む
        END IF
    END IF                                      'データを表示する
NEXT I
PRINT "何かキーを押してください。"              '表示が終了したら一時停止
DO
LOOP UNTIL INKEY$ <> ""                         '何か押すと次へ進む
END SUB
```

［データ入力のサブプロシージャ］

```
SUB NYUURYOKU (X, Y, ANNAI$, ONGAKUDATA$, L, KANZI)
HOZON$ = ONGAKUDATA$                            '渡されたデータを保存
IF KANZI THEN                                   '漢字を使用するならば
    Fep "ON"                                    '漢字入力モードON
END IF
LOCATE Y, X: PRINT ANNAI$; : INPUT "", ONGAKUDATA$          'データ入力する
IF KANZI THEN                                   '漢字を使用したら
    Fep "OFF"                                   '漢字入力モードをOFF
END IF
ONGAKUDATA$ = MID$(ONGAKUDATA$, 1, L)
                                        '入力したデータから必要な桁数を切り出す
IF ONGAKUDATA$ = "" THEN                        '何も入力されない場合は
    ONGAKUDATA$ = HOZON$                        '保存していたデータを使う
END IF                                          '入力時の表示を消して
LOCATE Y, X+LEN(ANNAI$):PRINT"                                   "
LOCATE Y, X + LEN(ANNAI$): PRINT ONGAKUDATA$    '切り出したデータを表示する
END SUB
```

［一覧表の表示のサブプロシージャ］

```
SUB ONGAKUITIRAN                                     '****一覧表の表示****
DIM ONGAKU AS ONGAKUTYPE                             '音楽ソフトのレコードの容器を作る
OPEN "B:ONGAKU.DAT" FOR RANDOM AS #1 LEN = LEN(ONGAKU)
                                                     '音楽ソフトファイルを開く
CLS                                                  '画面消去
PRINT "****音楽ソフトファイルの一覧表****"
PRINT
PRINT"番号 媒体 登録番号     曲     名          曲名読み     ";
                                              "歌  手  名       歌手名読み"
VIEW PRINT 5 TO 24
N = 0                                                '表示する行数のカウントをゼロ
FOR I = 1 TO LOF(1) ¥ LEN(ONGAKU)                    'レコードの数だけ繰り返す
    GET #1, I, ONGAKU                                'ファイルを読む
    IF ONGAKU.S THEN                                 'レコードが有効ならば
        PRINT  I;"  ";ONGAKU. BAITAI;" ";ONGAKU. BANGOU;" ";ONGAKU. KYOKU;" ";
        PRINT  ONGAKU. KYOKUYOMI;" ";ONGAKU. KASYU;" ";ONGAKU. KASYUYOMI
        N = N + 1                                    '画面に表示した行数を数える
        IF N MOD 15 = 0 THEN                         '１５行を表示したら止める
            PRINT : PRINT "何かキーを押してください。"
            DO
            LOOP UNTIL INKEY$ <> ""                  '何かキーを押したら次へ
            CLS
        END IF
    END IF
NEXT I
CLOSE #1                                             'ファイルを閉じる
PRINT : PRINT "何かキーを押してください。"
DO
LOOP UNTIL INKEY$ <> ""                              '何かキーを押したら次へ
VIEW PRINT 1 TO 25
END SUB
```

［索引簿(曲名·歌手名）ファイルの作成のサブプロシージャ］

```
SUB SAKUINBO
DIM ONGAKU AS ONGAKUTYPE                             '管理ファイルの容器を作る
DIM KYOKUSAKUIN(0 TO 1000) AS SAKUINTYPE
                                       '索引簿は0から1000までの番号をつけた容器
DIM KASYUSAKUIN(0 TO 1000) AS SAKUINTYPE
                                       '索引簿は0から1000までの番号をつけた容器
CLS                                                  '画面消去
PRINT "****索引簿（インデックスファイル）作成****"
PRINT
OPEN "B:ONGAKU.DAT" FOR RANDOM AS #1 LEN = LEN(ONGAKU) '管理ファイルを開く
'↓ファイルから曲名の読みを読み取る
FOR I = 1 TO LOF(1) ¥ LEN(ONGAKU)
```

```
    GET #1, I, ONGAKU                          '管理ファイルからデータを読み取る
    KYOKUSAKUIN(I).ONGAKUDATA = ONGAKU.KYOKUYOMI
                                               'データを索引簿へコピーする
    KASYUSAKUIN(I).ONGAKUDATA = ONGAKU.KASYUYOMI
NEXT I
SAKUINTUKURI KYOKUSAKUIN()                     '索引簿をつくる
SAKUINTUKURI KASYUSAKUIN()
INDKAKU KYOKUSAKUIN(), "B:KYOKU.IND"  '索引簿ファイルをフロッピーへ書き込む
INDKAKU KASYUSAKUIN(), "B:KASYU.IND"
CLOSE #1
END SUB
```

［曲名順、歌手名順の一覧表示のサブプロシージャ］

```
SUB SAKUINITIRAN
DIM ONGAKU AS ONGAKUTYPE                       '管理ファイルの容器を作る
DIM KYOKUSAKUIN(0 TO 1000) AS SAKUINTYPE
                                  '索引簿は0から1000までの番号をつけた容器
DIM KASYUSAKUIN(0 TO 1000) AS SAKUINTYPE
                                  '索引簿は0から1000までの番号をつけた容器
CLS                                            '画面消去
PRINT "****索引簿（インデックスファイル）作成****"
PRINT
OPEN "B:ONGAKU.DAT" FOR RANDOM AS #1 LEN = LEN(ONGAKU) '管理ファイルを開く
INDYOMIITIRAN KYOKUSAKUIN(), "B:KYOKU.IND",
                         "     ***** 音楽ソフト　曲名順一覧表 *****"
INDYOMIITIRAN KASYUSAKUIN(), "B:KASYU.IND",
                         "     ***** 音楽ソフト　歌手順一覧表 *****"
CLOSE #1
END SUB
```

［索引簿を表の形につくるサブプロシージャ］

```
SUB SAKUINTUKURI (SAKUIN() AS SAKUINTYPE)
DIM ONGAKU AS ONGAKUTYPE
SAKUIN(0).BANGOU = 1                           '索引簿の0番目の番号を1とする
                                               '次のデータが1番目にあることを示す
SAKUIN(1).BANGOU = 0                           '索引簿の1番目の番号を0とする
                           '索引番号のついた最後のデータであることを示す
FOR K = 2 TO LOF(1) ¥ LEN(ONGAKU)             '２番目からデータの数だけ繰り返す
    P = 0                                      'Pを0にしておく
    DO                                         '繰り返す
        Q = SAKUIN(P).BANGOU  '索引番号P番目のデータがもつ索引番号をQとする
        'P番目とQ番目のデータの間にK番目のデータが挿入されるときは
        IF SAKUIN(K).ONGAKUDATA < SAKUIN(Q).ONGAKUDATA THEN
                       'K番目の曲名の読みがQ番目の曲名の読みより小さければ
            SAKUIN(K).BANGOU = SAKUIN(P).BANGOU
```

```
                                    'P番目の番号をK番目の番号にコピーして
            SAKUIN(P).BANGOU = K            'P番目の番号へはKをコピーして
            EXIT DO                         '繰り返しを中止する
        END IF
        'K番目のデータが番号を付けたすべてのデータよりも大きいときは
        IF SAKUIN(Q).BANGOU = 0 THEN        'Q番目の番号が0ならば
            SAKUIN(Q).BANGOU = K            'KをQ番目の番号へコピーし
            SAKUIN(K).BANGOU = 0            'K番目の番号を0にし
            EXIT DO                         '繰り返しを中止する
        END IF
        P = Q                               'QをPへコピーする
    LOOP
NEXT K
END SUB
```

［訂正と削除のサブプロシージャ］

```
SUB TEISEI
DIM ONGAKU AS ONGAKUTYPE
DO
    PRINT
    PRINT "次のデータ:スペース　訂正:C　削除:D　-->"
    DO
        J$ = INKEY$                         '訂正と削除の選択
    LOOP UNTIL J$ = " " OR J$ = "C" OR J$ = "D"
    SELECT CASE J$
        CASE " "                            'スペースならば
            EXIT SUB                        'サブを抜けて戻る
        CASE "C"                            'Cならば
            IPUT "訂正するレコード番号　-->";J　'訂正するレコード番号の入力
            GET #1, J, ONGAKU
            INPUT
"訂正する項目　登録番号:1　曲名:2　曲名読み:3　歌手名:4　歌手名読み:5　-->"
                                    ,T              '訂正する項目の入力
            IF T > 1 THEN
                Fep "ON"                    '漢字入力モードON
            END IF
            SELECT CASE T                   '訂正項目を入力
                CASE 1
                    PRINT ONGAKU.BANGOU;
                    INPUT "-->", ONGAKU.BANGOU
                CASE 2
                    PRINT ONGAKU.KYOKU;
                    INPUT "-->",ONGAKU.KYOKU
                CASE 3
                    PRINT ONGAKU.KYOKUYOMI;
                    INPUT "-->", ONGAKU.KYOKUYOMI
                CASE 4
```

```
                    PRINT ONGAKU.KASYU;
                    INPUT "-->",ONGAKU.KASYU
                CASE 5
                    PRINT ONGAKU.KASYUYOMI;
                    INPUT "-->",ONGAKU.KASYUYOMI
            END SELECT
            IF T > 1 THEN
                Fep "OFF"                   '漢字入力モードＯＦＦ
            END IF
            PUT #1, J, ONGAKU               '訂正済みのレコードを書き込む
        CASE "D"
            INPUT "削除するレコード番号 -->";J '削除するレコード番号を入力
            GET #1, J, ONGAKU               '削除するレコードを読む
            ONGAKU.S = 0                    '削除するシルシをつける
            PUT #1, J, ONGAKU               '削除するレコードを書き込む
    END SELECT
LOOP
END SUB
```

［タイトル画面の表示のサブプロシージャ］

```
SUB TITLEHYOUZI
CLS
COLOR 10:   LOCATE  3,25: PRINT "***********************************"
            LOCATE  4,25: PRINT "*******                      *******"
            LOCATE  5,25: PRINT "*******  音楽ソフト管理     *******"
            LOCATE  6,25: PRINT "*******                      *******"
            LOCATE  7,25: PRINT "***********************************"
COLOR 6:    LOCATE  9,25: PRINT "1: データの入力"
            LOCATE 11,25: PRINT "2: 音楽ソフト一覧（入力順）"
            LOCATE 13,25: PRINT "3: データの検索・訂正・削除"
            LOCATE 15,25: PRINT "4: インデックスファイルの作成"
            LOCATE 17,25: PRINT "5: 曲名、歌手名の読み順のソフト一覧"
            LOCATE 19,25: PRINT "6: 終了"
COLOR 7:    LOCATE 21,25: PRINT "番号を入力してください。"
END SUB
```

第7章

売掛管理

入力技法とヘルプ画面のプログラミングについて学びます。

７．１　売掛管理の概要

ＣＯＢＯＬのプログラミングの学習では、必ずファイル処理の技法を学びます。中でも代表的な題材は、販売管理システムです。Ｑｕｉｃｋ　Ｂａｓｉｃでも同じようなファイル処理を取り扱ってもよいのですが、ここではあえて、Ｑｕｉｃｋ　Ｂａｓｉｃの特性を発揮できるようなプログラミングを紹介したいと思います。

販売管理は、厳密に検討すると仕入管理や在庫管理までも関連してその内容に包括され、プログラムが大きくなって説明や学習のためのプログラムとしては複雑になるので、ここでは単純化のために、売掛管理すなわち売掛と入金の記録をとって請求書を作成するまでの仕事のプログラムを作ることにしました。

まず、従前のファイル処理（シーケンシャルファイル）の概要は次のとおりです。

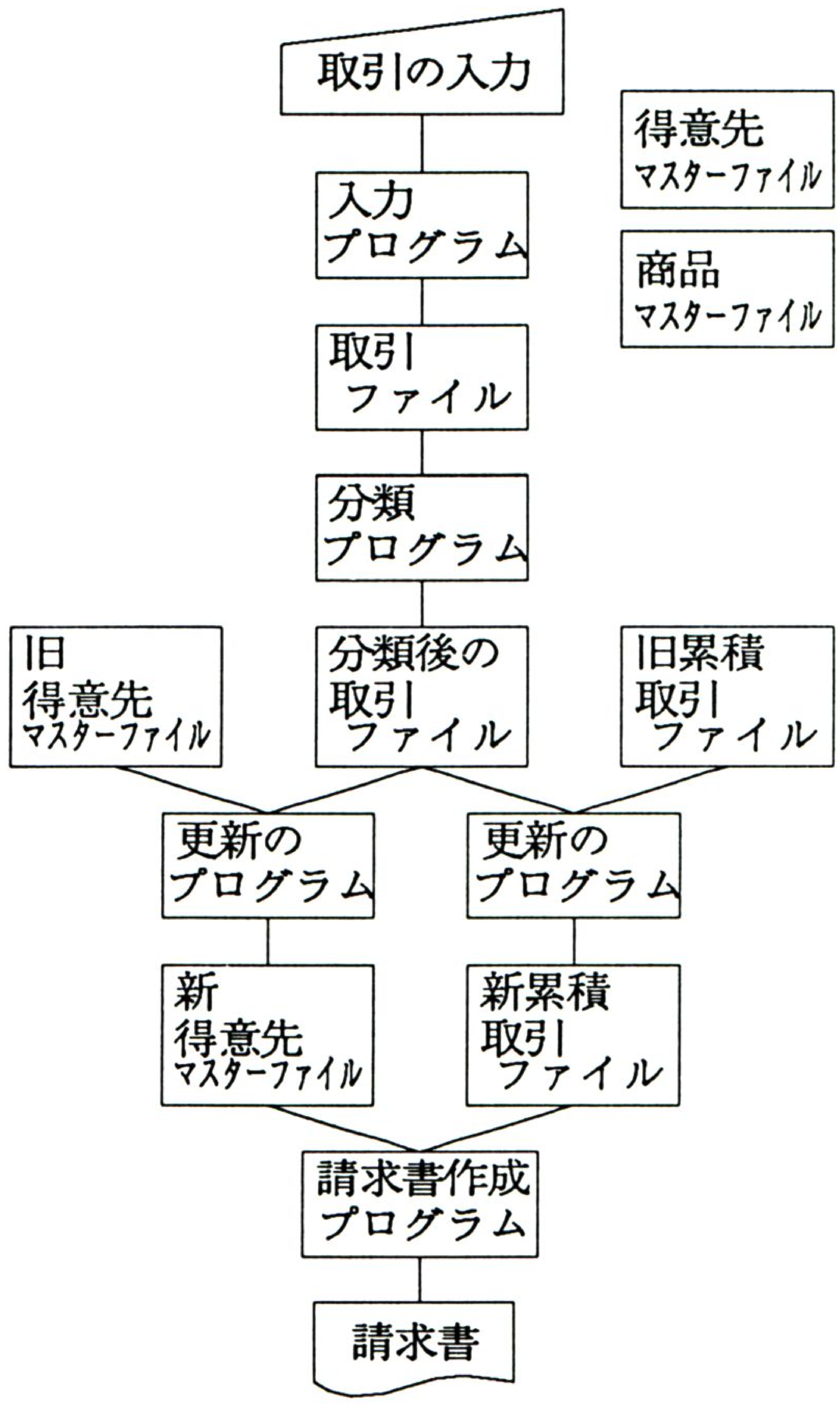
1. 毎日、商品マスターファイル及び得意先マスターファイルを参照しながら、取引（売掛、入金）を入力して取引ファイルを作ります。

2. 取引ファイルは、直ちに得意先番号順に並べ替え（分類）します。

3. 得意先マスターファイルを分類済みの取引ファイルによって更新して、新しい得意先マスターファイルを作ります。

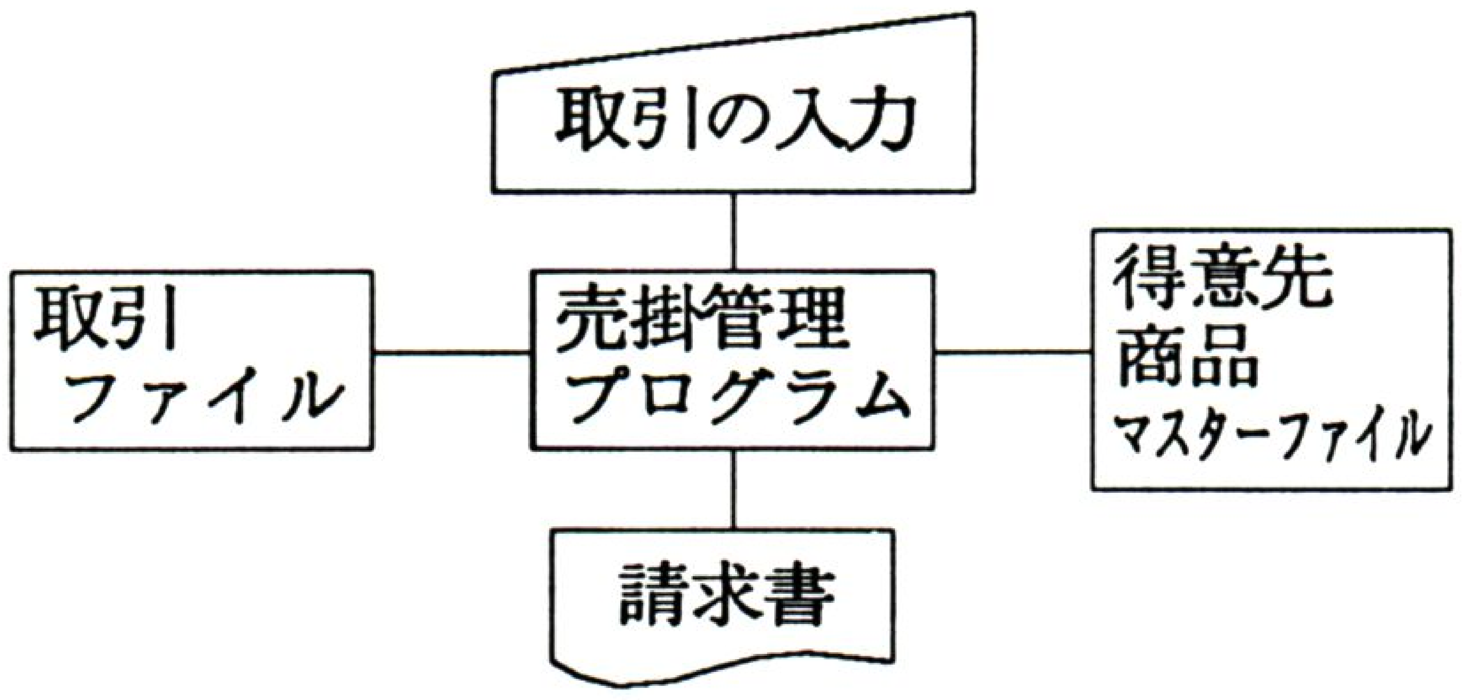
4. 累積取引ファイルを、分類済みの取引ファイルによって、更新して、新しい累積取引ファイルを作ります。

5. 得意先マスターファイル及び累積取引ファイルから、請求書を作ります。



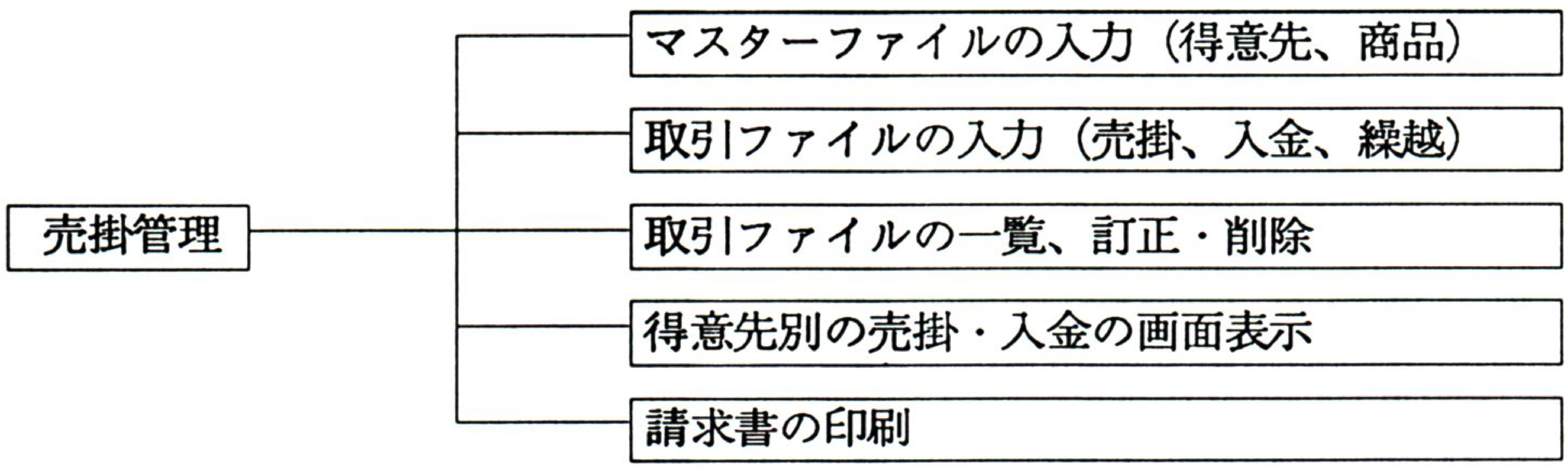
このシステムは、いわゆるバッチ処理（バッチ＝束。例えば１日分を一括して処理する方法）で、取引の１日分をまとめて入力し、これを得意先別に並べ替えて得意先マスターファイル及び累積の取引ファイルを更新し、最後に請求書を作成します。これがシーケンシャルファイル（順編成ファイル）の処理の方法の基本形です。

次は、いよいよＱｕｉｃｋ　Ｂａｓｉｃのランダムアクセスファイルを利用した売掛管理です。ファイル処理の概要は次のとおりです。



このように、ランダムアクセスファイルを利用すればファイルの更新や分類を毎日行う必要がないので、全体としてシンプルな構造になり、操作も容易になります。

売掛管理のプログラムは、次の５つのプログラムで構成します。



次に、プログラムを作成するためのデータを用意します。

１．マスターファイルのデータ

プログラムをシンプルなものにするために、得意先マスターファイルと商品マスターファイルをまとめて、マスターファイルとします。

項目は、得意先番号・商品番号と得意先名・商品名及び削除のシルシの３項目ですが、得意先番号・商品番号は、レコード番号と一致させることによって、省略できます。例えば、１０番斉藤商店のレコードを１０番のレコード番号の位置に書き込むことによって、１０番という番号の項目を省略できるのです。

また、１番から５０番までを得意先番号、５１番から１００番までを商品番号としました。

1	朝日商事	2	赤川商店	3	井上産業	4	伊藤店
5	宇野工業	6	宇佐見商事	7	上田産業	8	遠藤家電
9	小野寺商店	10	沖田建設	11	小山商事	12	金沢商店
13	柿沢電器	14	北山商店	15	北田商事	16	熊谷商事
17	熊田興業	18	黒田商店	19	栗田建設	20	倉田商店
21	駒田産業	22	近藤商事	23	今野商店	24	佐々木商事
25	斉藤家電	26	篠田電気	27	篠山産業	28	椎名商店
29	吹田商事	30	清野商事	31	園田興業	32	田中商店
33	津田建設	34	角田商事	35	戸田商店		

51	ラジオＡ	52	ラジオＢ	53	ラジオＣ	54	ＴＶ１２型
55	ＴＶ１４型	56	ＴＶ２０型	57	洗濯機Ａ型	58	洗濯機Ｂ型
59	冷蔵庫Ａ型	60	冷蔵庫Ｂ型	61	冷蔵庫Ｃ型	62	電子レンジ
63	ラジカセＡ	64	ラジカセＢ	65	ラジカセＣ	66	ＶＴＲ－Ａ
67	ＶＴＲ－Ｂ	68	ＶＴＲ－Ｃ				

2. 取引ファイルのデータ

前期繰越額

日付	得意先番号	金　額
90-10-01	12	12500
90-10-01	15	354000
90-10-01	20	85600
90-10-01	25	1000
90-10-01	30	23000

入金取引

日付	得意先番号	入金区分	金　額
90-10-06	12	1	100000
90-10-10	25	3	200000
		1	50000
90-10-12	15	2	500000
		3	100000
90-10-13	20	1	80000

売掛取引

日付	得意先番号	商品番号	数量	単価
90-10-05	12	57	5	53000
		59	3	89000
		55	10	87000
90-10-07	25	65	8	43000
90-10-08	15	52	12	21800
90-10-09	30	53	14	15800
90-10-10	12	52	5	23800
		59	3	89800
		62	2	128000
90-10-12	15	63	10	56800
		67	7	72600

7．2　マスターファイルの入力

マスターファイルは、データ処理の基本になるファイルで、この売掛管理システムでは得意先名と商品名とを登録します。一般には、得意先マスターファイルと商品マスターファイルは、別に用意しますが、便宜上、まとめてマスターファイルとしました。

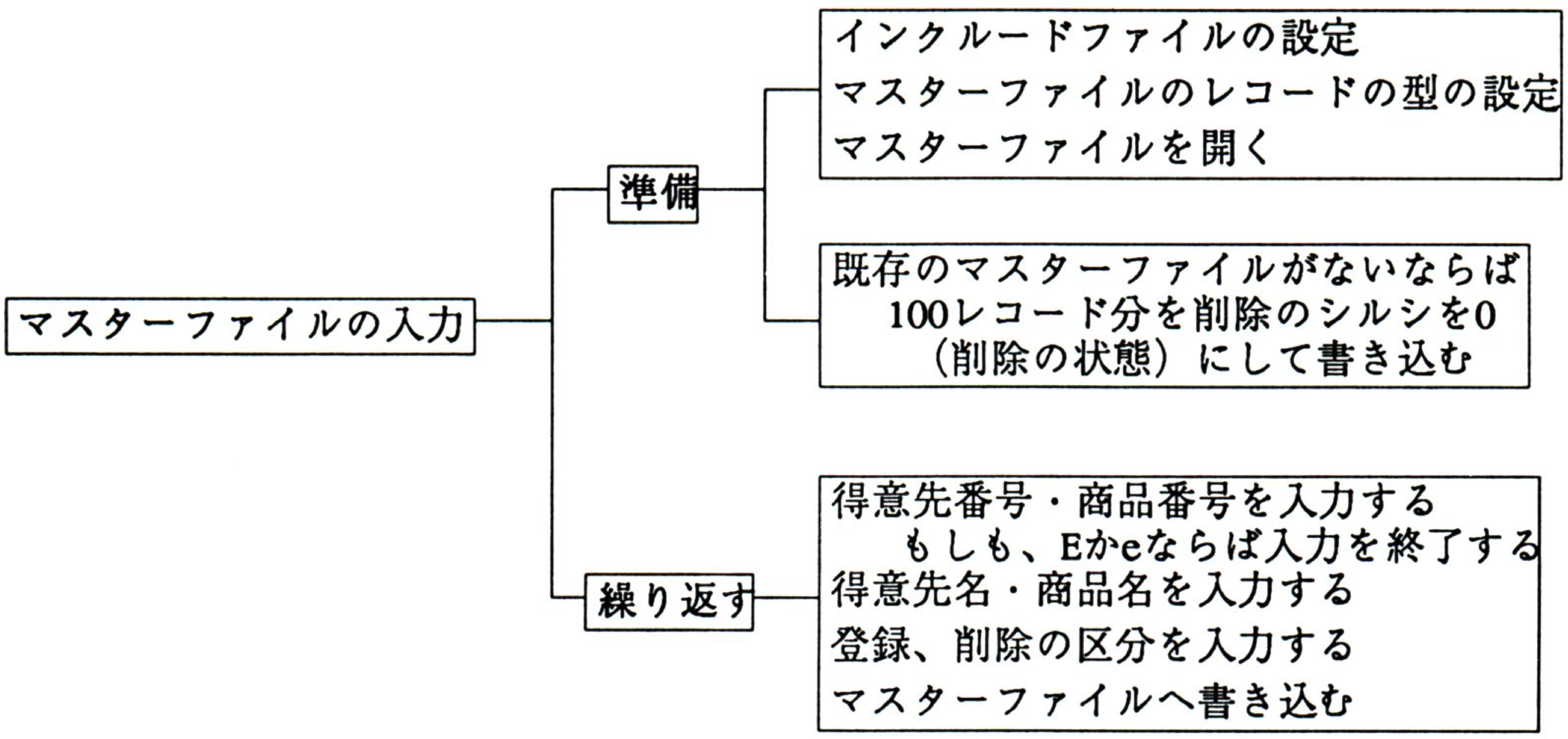
マスターファイルの項目は次のとおりです。

1． 得意先名・商品名

2． レコードの削除のシルシ

なお、得意先番号及び商品番号は、レコード番号の位置に得意先名・商品名を書き込むことによって表すことにしました。

マスターファイル入力のプログラムの構造は次のとおりです。




```
DEFINT A-Z
'売掛管理１        *****得意先・商品マスターファイルの登録・削除*****
'$INCLUDE: 'QB.BI'                        'インクルードファイルの設定
'$INCLUDE: 'GEN.BI'
TYPE MasterType                      'マスターファイルのレコードの型を決める
    Namae     AS STRING * 10              '得意先・商品の名前は全角で５字
    S         AS INTEGER                  '削除のシルシ
END TYPE
DIM Master AS MasterType           'マスターファイルのレコードはマスターの型
CLS
OPEN "B:Master.dat" FOR RANDOM AS #1 LEN = LEN(Master)
                                          'マスターファイルを開く
IF LOF(1) = 0 THEN                       'マスターファイルが存在しなければ
    Master.Namae = ""                     '名前を空にして
    Master.S = 0                          '削除の印を削除にして
```

```
    FOR I = 1 TO 100                            '100レコード分を
        PUT #1, I, Master                       '書き込む
    NEXT I
END IF
DO                                              '繰り返す
    PRINT                                       '改行
    INPUT "得意先番号・商品番号（終了:E） -->", B$              '番号の入力
    IF B$ = "E" OR B$ = "e" THEN                'もしEかeならば
        EXIT DO                                 '入力は終了
    END IF
    fep "on"                                    '漢字入力モードＯＮ
    INPUT "得意先名・商品名（終了:E） -->", M$       '得意先名・商品名の入力
    fep "off"                                   '漢字入力モードＯＦＦ
    Master.Namae = M$                           '名前をレコードの項目へ複写
    INPUT "登録:1　削除:2　-->", I               '登録、削除を選択
    SELECT CASE I
        CASE 1: Master.S = -1                   '登録ならば-1を
        CASE 2: Master.S = 0                    '削除ならば0をシルシとしてつける
    END SELECT
    PUT #1, VAL(B$), Master                     'マスターファイルへ書き込む
LOOP
CLOSE #1                                        'ファイルを閉じる
END
```

マスターファイルについては、特に訂正や削除のプログラムを用意しないで、登録、削除の区分を入力するようによって、登録と削除の機能をもたせるようにしました。

また、あらかじめ100レコード分のデータを削除のシルシを付けて入力しておきました。レコードがない状態で手入力を始めた場合、レコード番号が得意先番号を兼ねているので、不連続な得意先番号で入力するとまだ入力されていないレコードができることになり、したがってそのレコードの内容は不完全な状態になっている恐れがあるのです。最大のレコードの数は100レコードですから、1番から100番までのレコードをとにかく用意しておいたのです。

7．3　取引ファイルの入力

売掛、入金及び前月繰越の取引を入力するプログラムを作ります。
このプログラムは、特に次の点について工夫してあります。

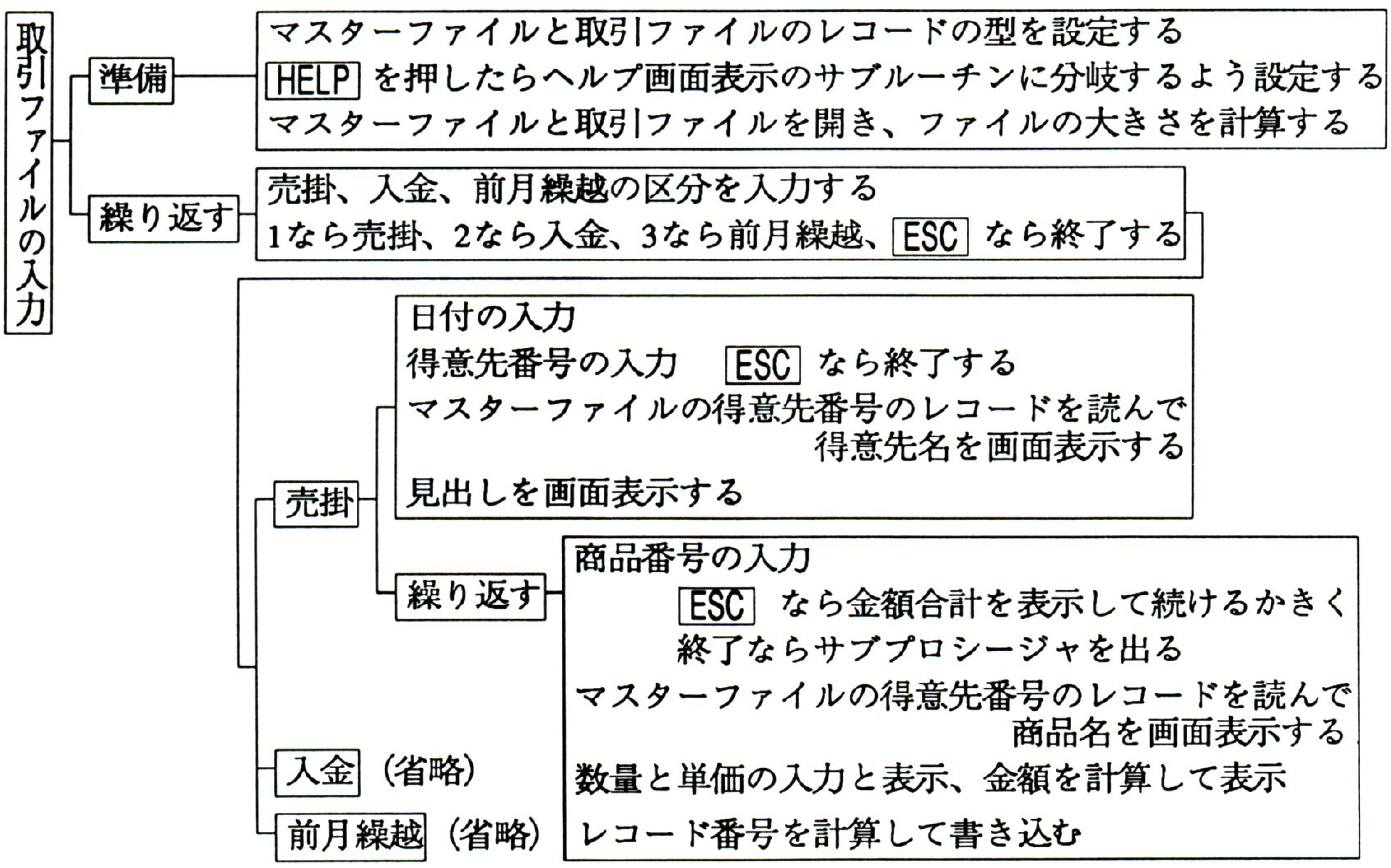
1. 7．1で触れたシーケンシャルファイルを使った処理では、これらの取引データはそれぞれ別のファイルとして入力し、さらにそれぞれを分類してマスターファイルを更新します。しかし、ここではこれらの取引をすべて一つのファイルへ入力し、分類しないでそのまま請求書の作成資料として利用できるようにしました。これによって、プログラムの量と操作する人の手間を減少できます。

2. 入力をわかりやすくするために、入力の画面表示を伝票形式にしました。

3. データの入力を終了するときは、［ESC］を押すことにしました。
　ただし、そのためには得意先番号と商品番号の入力部分のプログラムを変更しなければなりません。これまでのINPUTでは［ESC］を押されたことが判断できないので、キー入力を文字型1文字ずつで受け取って［ESC］が押されたか判断し、押されていなければその文字を連結して整数に変えて番号として使うようにしてあります。この処理をするサブプロシージャを用意しました。

4. 得意先番号と商品番号を入力するときに［HELP］を押すと、得意先番号と商品番号を案内するようにしました。

プログラムの構造は次のとおりです。



得意先番号および商品番号の入力のためのサブプロシージャの構造

- サブプロシージャ Nyuuryoku
  - 入力された文字を連結して渡すための変数をクリアする
    指示された位置へカーソル移動
  - 繰り返す
    - カーソルを点滅表示
      ヘルプの機能をＯＮ
      １文字入力
      ヘルプの機能をＯＦＦ
    - 数字ならば、文字に連結する
    - BS ならば、右の１文字を削除し、
      表示を消し、カーソルを戻す
    - ESC ならば、エスケープコードを文字へ入れて、
      表示を消し、サブプロシージャを出る
    - リターンならば、サブプロシージャを出る

ヘルプ画面を表示するためのサブルーチンの構造

- ヘルプ画面のサブルーチン
  - 準備
    - カーソルの現在位置を調べる
      画面の２０から２５行までを使用する
  - レコード番号1から100まで繰り返す
    - マスターファイルを読む
      有効なレコードならば、
      レコードを表示し、レコード数を数える
    - ４レコード分を表示したら、
      改行して、行数へ1を加える
      - ４行を表示したら、一時停止して、
        戻るか、次を見るか、入力する
        - 戻るならば、ヘルプ画面を消して、
          1行から25行までの画面を使用する
          元の位置へカーソル移動
          サブルーチンから出る
  - 後処理
    - 一時停止して、何かキーを押したら
      ヘルプ画面を消して、
      1行から25行までの画面を使用する
      元の位置へカーソル移動
      サブルーチンから出る

```
DEFINT A-Z
DECLARE SUB Nyuuryoku (y, x, S$)
DECLARE SUB Kurikosi (I, K$, S$)
DECLARE SUB Nyuukin (I, K$, S$)
DECLARE SUB Urikake (I, K$, S$)
'売掛管理２     *****売掛・入金ファイルの入力*****
'$INCLUDE: 'QB.BI'                    'インクルードファイルの設定
'$INCLUDE: 'GEN.BI'
TYPE Mastertype                       'マスターファイルのレコードの型
    Namae       AS STRING * 10        '得意先・商品の名前は全角で５字
    S           AS INTEGER            '削除のシルシ
END TYPE
TYPE Torihikitype                     '取引ファイルのレコードの型
    Kubun       AS STRING * 1         '区分（売掛:1  入金:2  繰越:3）
    hizuke      AS STRING * 8         '日付（yy-mm-dd）
    TokuiBangou AS INTEGER            '得意先番号
    SyouBangou  AS INTEGER            '商品番号
    Suuryou     AS INTEGER            '数量
    Tanka       AS LONG               '単価
    Kingaku     AS LONG               '金額
    S           AS INTEGER            '削除のシルシ
END TYPE
DIM Master      AS Mastertype         'マスターのレコードの型
DIM Torihiki    AS Torihikitype       '取引のレコードの型
ON KEY(31) GOSUB Helpsub:   '←   を押したらＨＥＬＰ画面を表示する

OPEN "B:Master.dat" FOR RANDOM AS #1 LEN = LEN(Master)
                                      'マスターファイルを開く
OPEN "B:Torihiki.dat" FOR RANDOM AS #2 LEN = LEN(Torihiki)
                                      '取引ファイルを開く
I = LOF(2) ¥ LEN(Torihiki)            '取引ファイルの大きさを計算
DO                                    '繰り返す
    CLS                               '画面消去
    LOCATE 10, 10
    PRINT "区分  売掛:1  入金:2  前月繰越:3  終了:ESC -->"
                                      '取引の区分を選択
    DO
        K$ = INKEY$                   '区分の入力
    LOOP UNTIL (K$ >= "1" AND K$ <= "3") OR K$ = CHR$(&H1B)
    DO
        SELECT CASE K$
            CASE "1": Urikake I, K$, S$                       '売掛
            CASE "2": Nyuukin I, K$, S$                       '入金
            CASE "3": Kurikosi I, K$, S$                      '繰越
            CASE IS = CHR$(&H1B): CLOSE #1: CLOSE #2: END     '終了
        END SELECT
    LOOP UNTIL S$ = CHR$(&H1B)        'S$がESCとなるまで繰り返す
LOOP
```

```
END
Helpsub:
    HX = POS(0): HY = CSRLIN                   'カーソルの現在位置を得る
    VIEW PRINT 20 TO 25                        '画面の20から25行までを使う
    HM = 1: HL = 0
    FOR H = 1 TO 100                  'マスターファイルの1から100番まで
        GET #1, H, Master                      'マスターファイルを読む
        IF Master.S THEN                       '有効なレコードならば
            PRINT USING "#### &          &  "; H; Master.Namae;
                                               'レコードを書く
            HM = HM + 1                        '書いたレコード数を数える
        END IF
        IF (HM - 1) MOD 4 = 0 THEN             '4レコード分表示したら
            PRINT                              '改行して
            HL = HL + 1                        '行数を計算し
            IF HL MOD 4 = 0 THEN               '4行表示したら
                PRINT "戻る:R  次を:N -->"            '一時停止して
                DO
                    Hin$ = INKEY$                     'RかNを入力
                LOOP UNTIL Hin$ = "R" OR Hin$ = "N"
                IF Hin$ = "R" THEN                    'Rならば
                    CLS                               '画面消去
                    VIEW PRINT 1 TO 25                '使用する画面を指定
                    LOCATE HY, HX                     '元の位置へカーソル移動
                    RETURN                            'サブプロシージャへ戻る
                END IF
            END IF
        END IF
    NEXT H
    PRINT:PRINT "何かキーを押してください。"          '表示を終了したら
    DO
    LOOP UNTIL INKEY$ <> ""                    '一時停止して
    CLS
    VIEW PRINT 1 TO 25                         '使用画面を指定
    LOCATE HY, HX                       (HELP) '元の位置へカーソル移動
RETURN
```

```
ON KEY(31) GOSUB Helpsub:
```

もしも、[HELP]を押したら、**Helpsub:**という名札の付いたプログラムの仕事をして、RETURNで戻るということです。

しかし、**KEY(31) GOSUB Helpsub:**の書いてある位置で押されるのを待っているのではありません。**KEY(31) ON**と **KEY(31) OFF**との間ならいつでも有効です。

KEY(31) ONと**KEY(31) OFF**は、**Nyuuryoku**というサブプロシージャにありますが、そこで役に立つのです。

KEY(31)の31という番号は、[HELP]を示します。他にもたとえばON KEY(1)ならば[f・1]を押せば、ということになります。

ON KEY(番号)の番号は次のように使うことができます。

[f・1]～[f・10]　：1～10

[↑]　：11　[←]　：12　[→]　：13　[↓]　：14

[STOP]　：30　[HELP]　：31

キーの割り当てはユーザ定義することができ、15から25がそのための番号です。

```
CASE IS = CHR$(&H1B): CLOSE #1: CLOSE #2: END
```

&Hは次に続く数が１６進数であることを示しますから、1Bは１６進数の1Bです。

CHR$()は()の中の数のアスキーコードに対応する文字をつくる関数で、アスキーコードの1Bは[ESC]を表しますから、この場合はK$に[ESC]が入力されていれば、ファイルを閉じて終了するということになります。

```
HX = POS(0): HY = CSRLIN
```

ヘルプ画面を表示して戻るときのカーソル位置を保存しています。

POS(0)は現在のカーソルの桁位置を、CSRLINは行位置を返す関数です。

[売掛取引の入力のサブプロシージャ]

```
SUB Urikake (I, K$, S$)
DIM Master   AS Mastertype                   'マスターレコードの型
DIM Torihiki  AS Torihikitype                '取引レコードの型
DIM Goukei AS LONG                           '合計は倍長の整数
CLS                                          '画面消去
Torihiki.Kubun = K$                          '区分を複写
Torihiki.S  = -1                             'レコードを有効に（削除は-1）
Goukei = 0                                   '合計を0にする
COLOR 2: LOCATE 1, 30: PRINT "****売掛の入力****"
COLOR 3: LOCATE 3, 45: PRINT "年月日   ";
COLOR 7: INPUT "", Torihiki.hizuke           '日付を入力
DO                                           '繰り返す
    COLOR 3: LOCATE 5, 8: PRINT "得意先番号(終了:ESC)";
    COLOR 7: Nyuuryoku 5, 30, S$             '得意先番号を入力
    IF S$ = CHR$(&H1B) THEN                  'ＥＳＣが押されれば
        EXIT SUB                             'サブプロシージャを出る
    ELSE                                     'そうでなければ
        Torihiki.TokuiBangou = VAL(S$)       '得意先番号を画面表示
        LOCATE 5, 30
        PRINT USING "#######"; Torihiki.TokuiBangou;
```

```
    END IF
    GET #1, Torihiki.TokuiBangou, Master 'マスターファイルを読む
LOOP UNTIL Master.S                      '有効なレコードを読むまで
COLOR 3: LOCATE 5, 40: PRINT "得意先名   ";
COLOR 7: PRINT Master.Namae              '得意先名を表示する
COLOR 3
LOCATE 7, 8
PRINT " 商品番号(終了:ESC)  商品名        数量      単  価      金      額"
COLOR 7
y = 9                                    '画面の9行目を設定
DO
    DO
        DO
            LOCATE y, 15: PRINT "        ";
            Nyuuryoku y, 15, S$          '商品番号を入力
            IF S$ = CHR$(&H1B) THEN      'もしもESCが入力されれば
                LOCATE y, 15: PRINT "        ";
                COLOR 3: LOCATE y, 53: PRINT "合計";
                COLOR 7: PRINT USING "####,###,###"; Goukei
                                         '合計を表示して
                LOCATE 18, 1
                PRINT "売掛の入力を続ける:何かキーを押す。 終了:ESC -->"
                DO                       '一時停止する
                    S$ = INKEY$
                LOOP UNTIL S$ <> ""
                EXIT SUB                 'サブプロシージャを出る
            END IF
            Torihiki.SyouBangou = VAL(S$)   '商品番号を取引レコードへ複写
            LOCATE y, 15: PRINT USING "####";Torihiki.SyouBangou;
                                         '商品番号を表示
        LOOP UNTIL Torihiki.SyouBangou>0 AND Torihiki.SyouBangou<=100
        GET #1, Torihiki.SyouBangou, Master        'マスターファイルを読む
    LOOP UNTIL Master.S
    LOCATE y, 28: PRINT Master.Namae;                              '商品名を表示
    LOCATE y, 40: INPUT "", Torihiki.Suuryou                        '数量を入力
    LOCATE y, 40: PRINT USING "#####"; Torihiki.Suuryou;            '数量を表示
    LOCATE y, 46: INPUT "", Torihiki.Tanka                          '単価を入力
    LOCATE y, 46: PRINT USING "######,###"; Torihiki.Tanka;         '単価を表示
    Torihiki.Kingaku = Torihiki.Suuryou * Torihiki.Tanka           '金額を計算
    PRINT USING "#####,###,###"; Torihiki.Kingaku                  '金額を表示
    I = I + 1                            'レコード番号を計算
    PUT #2, I, Torihiki                 '取引レコードをファイルへ書き込む
    Goukei = Goukei + Torihiki.Kingaku   '合計額の計算
    y = y + 2                            '画面表示の行位置の計算
LOOP
END SUB
```

[入金のサブプロシージャ]

```
SUB Nyuukin (I, K$, S$)
DIM Master   AS Mastertype                  'マスターレコードの設定
DIM Torihiki  AS Torihikitype               '取引レコードの設定
DIM Goukei AS LONG                          '合計を倍長整数と設定
CLS
Torihiki.Kubun = K$                         '区分を複写
 Torihiki.S  = -1                           'レコードを有効とする(削除は0)
Goukei = 0
COLOR 2: LOCATE 1, 25: PRINT "****売掛入金の入力****"
COLOR 3: LOCATE 3, 45: PRINT "年月日   ";
COLOR 7: INPUT "", Torihiki.hizuke          '日付を入力
DO
    COLOR 3: LOCATE 5, 8: PRINT "得意先番号(終了:ESC)";
    COLOR 7: Nyuuryoku 5, 30, S$            '得意先番号を入力
    IF S$ = CHR$(&H1B) THEN
        EXIT SUB
    ELSE
        Torihiki.TokuiBangou = VAL(S$)
        LOCATE 5, 30: PRINT USING "#######"; Torihiki.TokuiBangou;
    END IF
    GET #1, Torihiki.TokuiBangou, Master 'マスターファイルを読む
LOOP UNTIL Master.S
COLOR 3: LOCATE 5, 40: PRINT "得意先名   ";
COLOR 7: PRINT Master.Namae                 '得意先名を表示
COLOR 3
LOCATE 7, 14
PRINT "入金区分(現金小切手:1 振込:2 手形:3  終了:ESC)   金    額"
COLOR 7
y = 9                                       '画面の行位置を設定
DO
    DO
        LOCATE y, 19: PRINT "      ";
        Nyuuryoku y, 19, S$                 '入金区分を入力
        IF S$ = CHR$(&H1B) THEN             'もしもESCならば
            LOCATE y, 19: PRINT "      ";
            COLOR 3: LOCATE y, 56: PRINT "合計";              '合計を表示し
            COLOR 7: PRINT USING "####,###,###"; Goukei
            LOCATE 19, 1
            PRINT "入金の入力を続ける:何かキーを押す  終了:ESC -->"
            DO                              '一時停止する
                S$ = INKEY$
            LOOP UNTIL S$ <> ""
            EXIT SUB
        END IF
        Torihiki.SyouBangou = VAL(S$)       '入金区分を商品番号へ複写
        LOCATE y, 19: PRINT USING "####"; Torihiki.SyouBangou;
                                            '入金区分番号を表示
```

```
    LOOP UNTIL Torihiki.SyouBangou >= 1 AND Torihiki.SyouBangou <= 3
    SELECT CASE Torihiki.SyouBangou
        CASE 1: Kubun$ = "現金小切手"
        CASE 2: Kubun$ = "振込"
        CASE 3: Kubun$ = "手形"
    END SELECT
    LOCATE y, 26: PRINT USING "        &          &"; Kubun$;  '入金区分を表示
    LOCATE y, 60: INPUT "", Torihiki.Kingaku                    '金額を入力
    LOCATE y, 60: PRINT USING "####,###,###"; Torihiki.Kingaku
                                                                '金額を表示
    I = I + 1                                    'レコード番号を計算
    PUT #2, I, Torihiki                          '取引レコードを書き込む
    Goukei = Goukei + Torihiki.Kingaku           '合計額を計算
    y = y + 2                                    '画面の次の行位置を計算
LOOP
END SUB
```

［繰越の入力のサブプロシージャ］

```
SUB Kurikosi (I, K$, S$)
DIM Master   AS Mastertype                       'マスターレコードの設定
DIM Torihiki AS Torihikitype                     '取引レコードの設定
CLS                                              '画面消去
Torihiki.Kubun = K$                              '区分を複写
Torihiki.S = -1                                  'レコードが有効のシルシ
Goukei = 0
COLOR 2: LOCATE 5, 20: PRINT "****売掛の前月繰越額の入力****"
COLOR 3: LOCATE 7, 45: PRINT "年月日    ";
COLOR 7: INPUT "", Torihiki.hizuke               '日付の入力
DO
    DO
        LOCATE 9, 30: PRINT "            ";
        COLOR 3: LOCATE 9, 8: PRINT "得意先番号(終了:ESC)";
        COLOR 7: Nyuuryoku 9, 30, S$                 '得意先番号を入力
        IF S$ = CHR$(&H1B) THEN
            EXIT SUB
        ELSE
            Torihiki.TokuiBangou = VAL(S$)
            LOCATE 9, 30: PRINT USING "#######";Torihiki.TokuiBangou;
        END IF
        GET #1, Torihiki.TokuiBangou, Master         'マスターファイルを読む
    LOOP UNTIL Master.S                          'レコードがなければ再入力
    COLOR 3: LOCATE 9, 40: PRINT "得意先名    ";
    COLOR 7: PRINT Master.Namae                      '得意先名を表示
    COLOR 3: LOCATE 11, 55: PRINT "金      額"
    COLOR 7: y = 13                                  '行位置を設定
    LOCATE y, 50: INPUT "", Torihiki.Kingaku         '金額を入力
    LOCATE y, 50: PRINT USING "#####,###,###"; Torihiki.Kingaku
```

```
                                        '金額を表示
    I = I + 1                           'レコード番号を加える
    PUT #2, I, Torihiki                 '取引レコードを書き込む
LOOP
END SUB
```

[入力のためのサブプロシージャ]

```
SUB Nyuuryoku (y, x, S$)
S$ = ""                              '入力された文字を連結してまとめる文字
LOCATE y, x                          '指示された位置へカーソル移動
DO
    COLOR 23: PRINT "_"; CHR$(&H1D); : COLOR 7   '_を点滅して表示
    KEY(31) ON                       'ヘルプの機能をＯＮにする
    DO
        K$ = INKEY$                  '１文字を入力
    LOOP UNTIL K$ <> ""
    KEY(31) OFF                      'ヘルプの機能をＯＦＦにする
    SELECT CASE K$
        CASE "0", "1", "2", "3", "4", "5", "6", "7", "8", "9"
            S$ = S$ + K$             'K$が数ならばS$へ連結する
            PRINT K$;                'K$を表示
        CASE CHR$(&H8)               'バックスペースキーが押されたら
            S$ = MID$(S$, 1, LEN(S$) - 1)      '右の１文字を削る
            PRINT CHR$(&H1D); "  "; CHR$(&H1D); CHR$(&H1D);
                                     '表示を消し、カーソルを戻す
        CASE CHR$(&H1B)              'エスケープキーが押されれば
            S$ = K$                  'エスケープコードをS$へ送って
            LOCATE y, x: PRINT "        ";     '表示を消し
            EXIT SUB                 'サブプロシージャを出る
        CASE CHR$(&HD)               'リターンキーが押されれば
            EXIT SUB                 'サブプロシージャを出る
    END SELECT
LOOP
END SUB
```

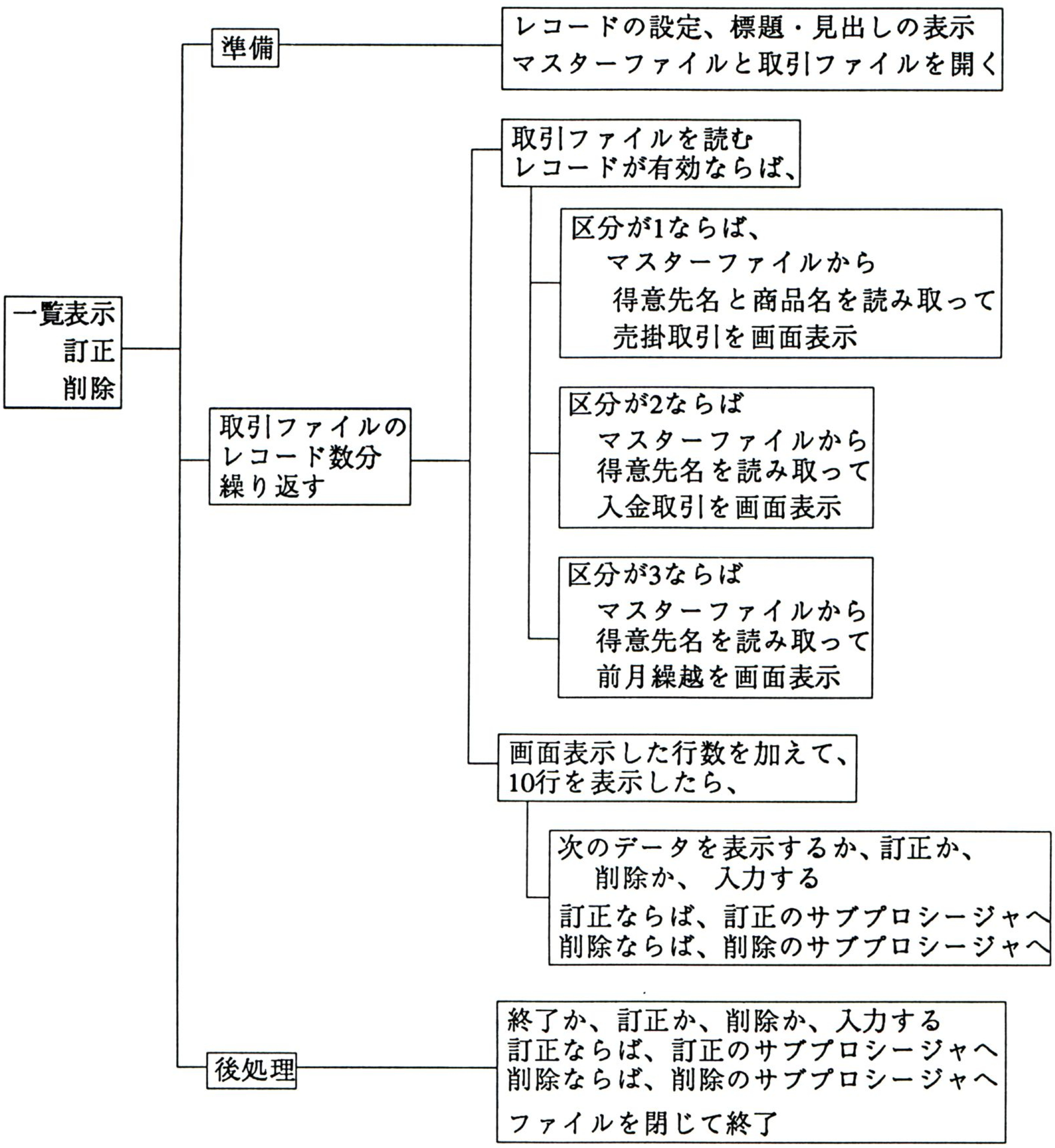
```
                         ****売掛の入力****
                                             年月日    90-10-05
    得意先番号（終了：ESC）        12    得意先名   金沢商店
    商品番号（終了：ESC）    商品名        数量       単  価      金    額
          57                洗濯機Ａ型       5        53,000      265,000
          59                冷蔵庫Ａ型       3        89,000      267,000
          55                ＴＶ１４型      10        87,000      870,000
                                                     合計    1,402,000
売掛の入力を続ける：何かキーを押す。  終了：ESC -->
```

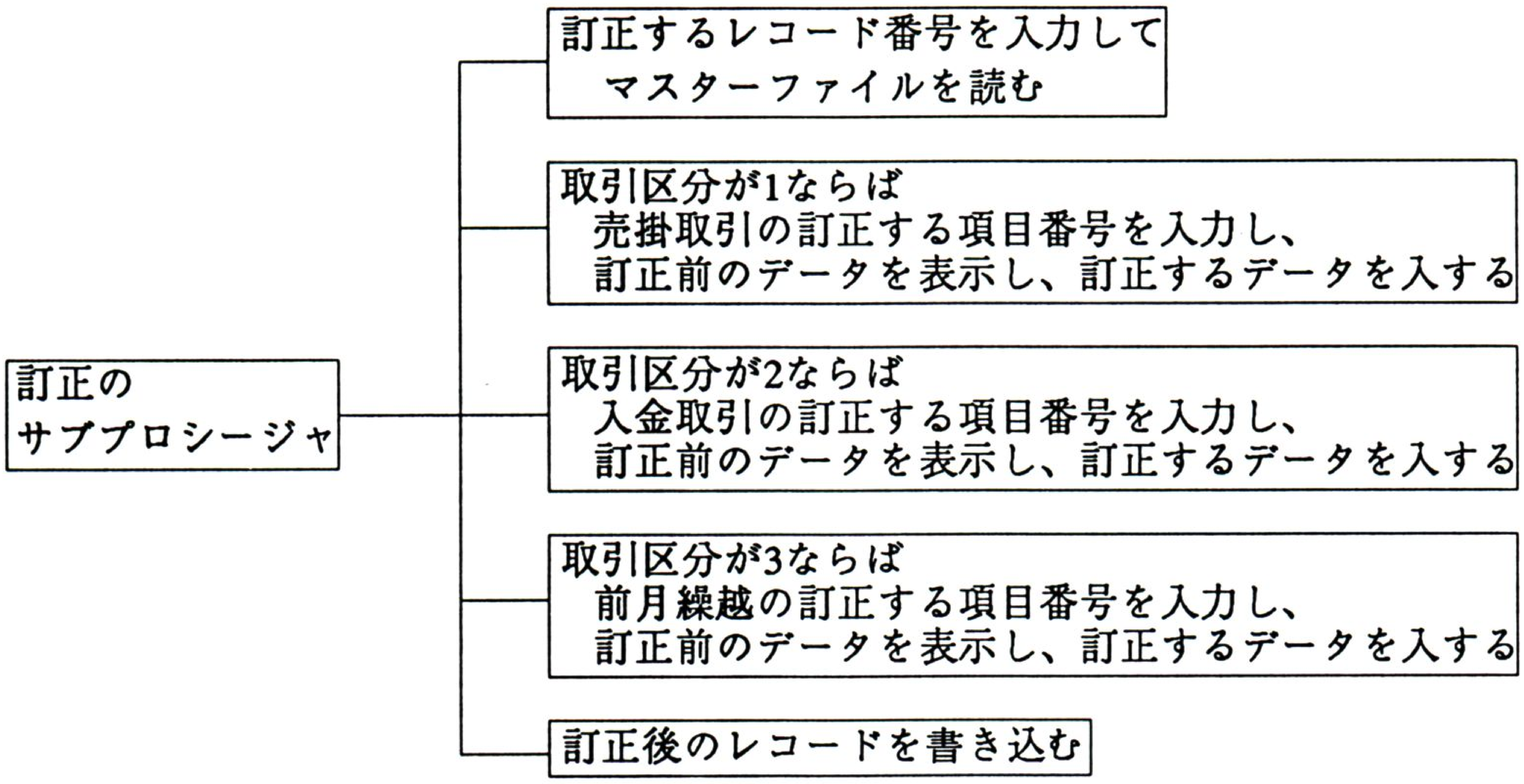
7．4　一覧表示・訂正・削除

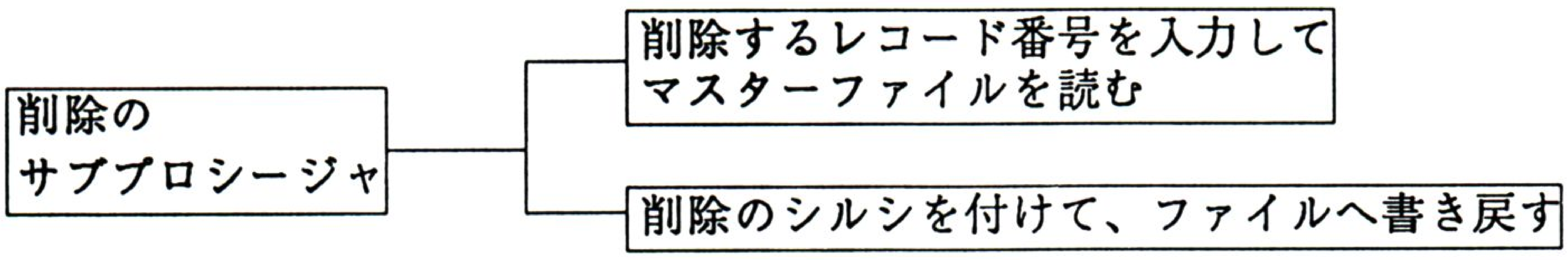
データの入力ができれば、それを画面に一覧表示したり、訂正や削除をする機能が必要となります。

そこで、これらの機能を一つのプログラムにまとめて作りました。

プログラムの構造は次のとおりです。








```
DECLARE SUB Sakuzyo ()
DECLARE SUB Teisei  ()
DEFINT A-Z
'売掛管理３   *****売掛・入金ファイルの画面表示、訂正、削除*****
'$INCLUDE: 'QB.BI'                         'インクルードファイルの設定
'$INCLUDE: 'GEN.BI'
TYPE Mastertype                            'マスターファイルのレコードの型
     Namae        AS STRING * 10           '得意先・商品の名前は全角で５字
     S            AS INTEGER               '削除のシルシ
END TYPE
TYPE Torihikitype                          '取引ファイルのレコードの型
     Kubun        AS STRING * 1            '取引区分
     Hizuke       AS STRING * 8            '日付（yy-mm-dd）
     TokuiBangou  AS INTEGER               '得意先番号
     SyouBangou   AS INTEGER               '商品番号
     Suuryou      AS INTEGER               '数量
     Tanka        AS LONG                  '単価
     Kingaku      AS LONG                  '金額
     S            AS INTEGER               '削除のシルシ
END TYPE
DIM Master        AS Mastertype            'マスターレコードの設定
DIM Torihiki      AS Torihikitype          '取引レコードの設定
CLS
```

```
PRINT "      *****売掛・入金ファイルの画面表示、訂正、削除*****"
PRINT
PRINT "ﾚｺｰﾄﾞ 区分 日付 得番 得意先名  品番   商品名     数量      ";
                                                    "単 価       金 額"
VIEW PRINT 4 TO 25
OPEN "B:Master.dat" FOR RANDOM AS #1 LEN = LEN(Master)
                                          'マスターファイルを開く
OPEN "B:Torihiki.dat" FOR RANDOM AS #2 LEN = LEN(Torihiki)
                                          '取引ファイルを開く
N = LOF(2) ¥ LEN(Torihiki)                'ファイルの大きさを計算
L = 0                                     '行数Lをゼロとする
FOR I = 1 TO N
    GET #2, I, Torihiki                   '取引ファイルを読む
    IF Torihiki.S THEN
        SELECT CASE Torihiki.Kubun
            CASE "1"                      '区分が1ならば
                PRINT I; " ";Torihiki.Kubun; "  ";Torihiki.Hizuke;
                GET #1, Torihiki.TokuiBangou, Master
                PRINT USING
                    "#### &        &";Torihiki.TokuiBangou;Master.Namae;
                GET #1, Torihiki.SyouBangou, Master
                PRINT USING
                    "#### &        &";Torihiki.SyouBangou; Master.Namae;
                PRINT USING "######"; Torihiki.Suuryou;
                PRINT USING "#######,###"; Torihiki.Tanka;
                PRINT USING "####,###,###"; Torihiki.Kingaku
            CASE "2"                      '区分が2ならば
                PRINT I; " ";Torihiki.Kubun; "  ";Torihiki.Hizuke;
                GET #1, Torihiki.TokuiBangou, Master
                PRINT USING
                    "#### &        &";Torihiki.TokuiBangou;Master.Namae;
                PRINT USING "####"; Torihiki.SyouBangou;
                SELECT CASE Torihiki.SyouBangou     '区分番号で区分を複写
                    CASE 1: Kubun$ = "現金小切手"
                    CASE 2: Kubun$ = "振込"
                    CASE 3: Kubun$ = "手形"
                END SELECT
                PRINT USING "  &        &"; Kubun$;
                PRINT USING
                        "                     ####,###,###";Torihiki.Kingaku
            CASE "3"                      '区分が3ならば
                PRINT I; " ";Torihiki.Kubun;"  ";Torihiki.Hizuke;
                GET #1, Torihiki.TokuiBangou, Master
                PRINT USING
                  "#### &        &";Torihiki.TokuiBangou; Master.Namae;
                PRINT "       前月繰越    ";
                PRINT USING
                        "                     ####,###,###";Torihiki.Kingaku
        END SELECT
```

```
        L = L + 1                               '表示した行数を計算
        IF L MOD 10 = 0 THEN                    '10行ごとに
            DO                                  '一時停止して
                PRINT "次のデータ:スペース  訂正:C  削除:D  -->"
                DO
                    T$ = INKEY$                 '次か、訂正か、削除か記号を入力
                LOOP UNTIL T$ = " " OR T$ = "C" OR T$ = "D"
                SELECT CASE T$
                    CASE "C": Teisei            'Cならば訂正
                    CASE "D": Sakuzyo           'Dならば削除
                END SELECT
            LOOP UNTIL T$ = " "
        END IF
    END IF
NEXT I
DO
    PRINT "終了:スペース  訂正:C  削除:D  -->"
    DO
        T$ = INKEY$                             '終了、訂正か、削除かを入力
    LOOP UNTIL T$ = " " OR T$ = "C" OR T$ = "D"
    SELECT CASE T$
        CASE "C": Teisei                        'Cならば訂正
        CASE "D": Sakuzyo                       'Dならば削除
    END SELECT
LOOP UNTIL T$ = " "
CLOSE #1, #2                                    'ファイルを閉じる
END
```

[訂正のサブプロシージャ]

```
SUB Teisei                                      'サブプロシージャ  訂正
DIM Master      AS Mastertype
DIM Torihiki    AS Torihikitype
PRINT : INPUT "訂正するレコード番号を入力してください。-->", R
GET #2, R, Torihiki
PRINT "訂正する項目を選択して番号を入力してください。"
SELECT CASE Torihiki.Kubun
    CASE "1"                                    '売掛取引の訂正
        INPUT "日付:1  得意先番号:2  商品番号:3  数量:4  単価:5  -->",B$
        SELECT CASE B$
            CASE "1": PRINT Torihiki.Hizuke; "-->";
                INPUT "", Torihiki.Hizuke
            CASE "2": PRINT Torihiki.TokuiBangou; "-->";
                INPUT "", Torihiki.TokuiBangou
            CASE "3": PRINT Torihiki.SyouBangou; "-->";
                INPUT "", Torihiki.SyouBangou
            CASE "4": PRINT Torihiki.Suuryou; "-->";
                INPUT "", Torihiki.Suuryou
```

```
            Torihiki.Kingaku=Torihiki.Suuryou*Torihiki.Tanka
        CASE "5": PRINT Torihiki.Tanka; "-->";
            INPUT "", Torihiki.Tanka
            Torihiki.Kingaku=Torihiki.Suuryou*Torihiki.Tanka
    END SELECT
  CASE "2"                                    '入金取引の訂正
    INPUT "日付:1  得意先番号:2  入金区分番号:3  入金額:4  -->", B$
    SELECT CASE B$
        CASE "1": PRINT Torihiki.Hizuke; "-->";
            INPUT "", Torihiki.Hizuke
        CASE "2": PRINT Torihiki.TokuiBangou; "-->";
            INPUT "", Torihiki.TokuiBangou
        CASE "3": PRINT Torihiki.SyouBangou; "-->";
            INPUT "", Torihiki.SyouBangou
        CASE "4": PRINT Torihiki.Kingaku; "-->";
            INPUT "", Torihiki.Kingaku
    END SELECT
  CASE "3"                                    '前月繰越の訂正
    INPUT "日付:1  得意先番号:2  入金額:3  -->", B$
    SELECT CASE B$
        CASE "1": PRINT Torihiki.Hizuke; "-->";
            INPUT "", Torihiki.Hizuke
        CASE "2": PRINT Torihiki.TokuiBangou; "-->";
            INPUT "", Torihiki.TokuiBangou
        CASE "3": PRINT Torihiki.Kingaku; "-->";
            INPUT "", Torihiki.Kingaku
    END SELECT
END SELECT
PUT #2, R, Torihiki                           '訂正済みのレコードの書き込み
END SUB
```

[削除のサブプロシージャ]

```
SUB Sakuzyo                                   'サブプロシージャ  削除
DIM Torihiki AS Torihikitype
PRINT : INPUT "削除するレコード番号を入力してください。-->", R
GET #2, R, Torihiki                           '削除するレコードを読む
Torihiki.S =0                                 '削除するシルシをつける
PUT #2, R, Torihiki                           '削除するレコードを書き込む
END SUB
```

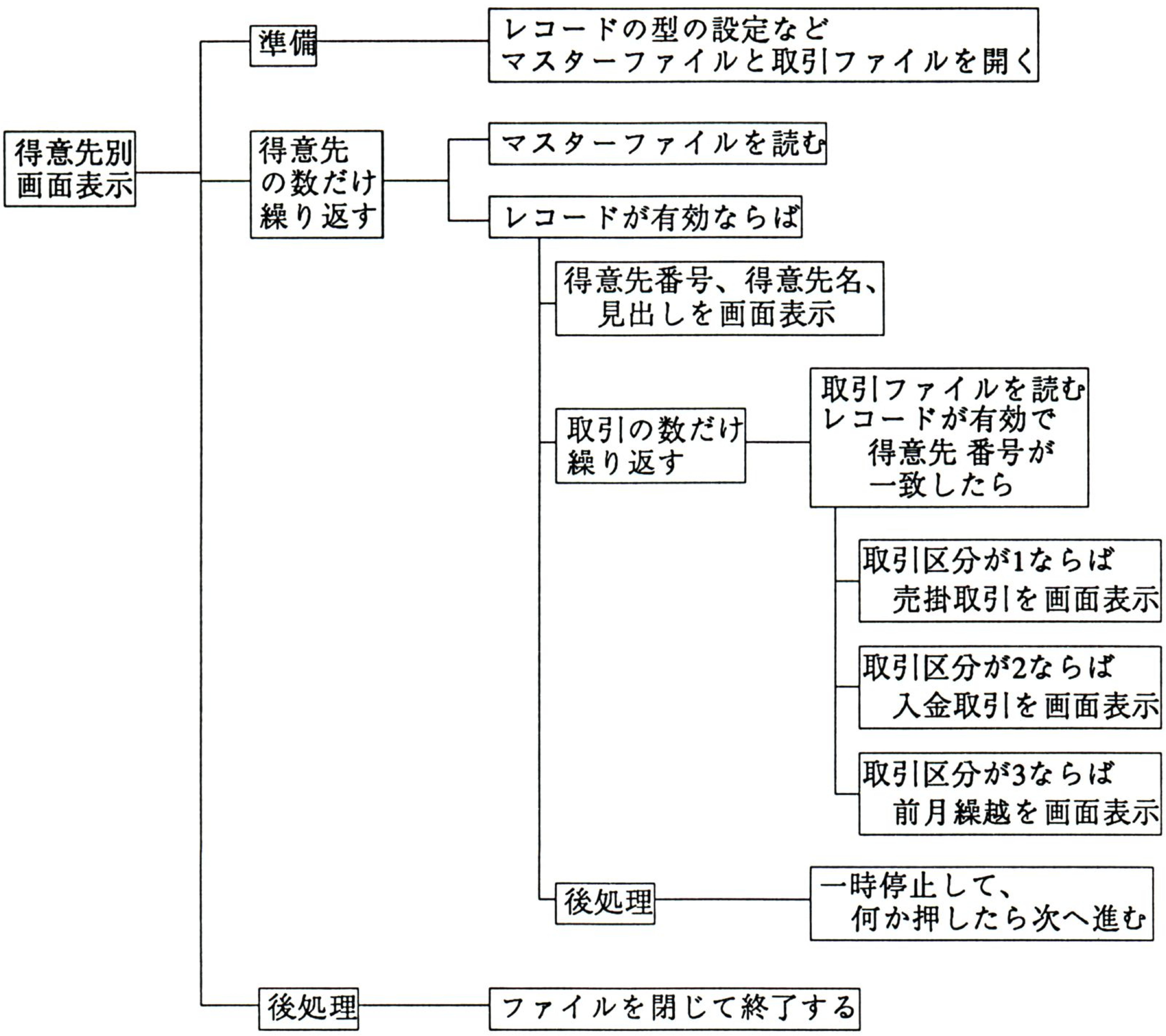
7．5　得意先別表示と請求書の印刷

取引の入力ができましたから、次は得意先別の売掛関係の取引をまとめて表示する機能と、月末に請求書の印刷をする機能を作ります。

これらのプログラムは、画面に表示することとプリンターに印刷することの違いがあるだけで、基本的には同じなのです。

そこで、はじめに画面表示のプログラムを作り、次に、画面に表示するPRINTを印刷のLPRINTへ変更し、請求書という標題を加えればよいのです。また、請求書は1件1枚として改ページをしました。改ページは、LPRINT CHR$(12)で実行できます。

プログラムの構造は次のとおりです。



請求書の印刷のプログラムの構造は、上記にほぼ同じなので省略します。

```
DEFINT A-Z
'売掛管理４      *****得意先別売掛入金取引の画面表示*****
TYPE Mastertype                              'マスターファイルのレコードの型
     Namae       AS STRING * 10              '得意先・商品の名前は全角で５字
     S           AS INTEGER                  '削除のシルシ
END TYPE
TYPE Torihikitype                            '取引ファイルのレコードの型
     kubun       AS STRING * 1               '取引の区分
     Hizuke      AS STRING * 8               '日付
     TokuiBangou AS INTEGER                  '得意先番号
     Syoubangou  AS INTEGER                  '商品番号
     Suuryou     AS INTEGER                  '数量
     Tanka       AS LONG                     '単価
     Kingaku     AS LONG                     '金額
     S           AS INTEGER                  '削除のシルシ
END TYPE
DIM Master       AS Mastertype               'マスターレコードの設定
DIM Torihiki     AS Torihikitype             '取引レコードの設定
DIM Zandaka      AS LONG                     '残高は倍長整数
CLS
PRINT "      *****得意先別売掛入金取引の画面表示*****"
PRINT
VIEW PRINT 4 TO 25
OPEN "B:Master.dat" FOR RANDOM AS #1 LEN = LEN(Master)
OPEN "B:Torihiki.dat" FOR RANDOM AS #2 LEN = LEN(Torihiki)
FOR I = 1 TO 50          'マスターファイルの１から５０までのレコードを読む
     GET #1, I, Master                       'マスターファイルを読む
     IF Master.S THEN
          PRINT : PRINT USING "    ###    &          &";I;Master.Namae
          PRINT        "-------------------------------------------------
                    <前行続き>    -------------------------"
          PRINT " ﾚｺｰﾄﾞ  日付  商品番号  商品名        数量    単  価 ";
                    <前行続き>"    金    額    残    高"
          PRINT        "-------------------------------------------------
                    <前行続き>    -------------------------"
          J = LOF(2) ¥ LEN(Torihiki)         '取引ファイルの大きさを計算
          Zandaka = 0                        '残高をゼロとする
          L = 8                              '取引データを表示した行数
          FOR K = 1 TO J
               GET #2, K, Torihiki           '取引ファイルを読む
               IF Torihiki.S AND I = Torihiki.TokuiBangou THEN
                    SELECT CASE Torihiki.kubun
                         CASE "1"            '売掛取引の場合
                              PRINT USING "####"; K;             'レコード番号
                              PRINT USING "  &       &  "; Torihiki.Hizuke;' 日付
                              PRINT USING "#####"; Torihiki.Syoubangou;
                                                                  '商品番号
                              GET #1, Torihiki.Syoubangou, Master
                                                     'マスターファイルを読む
                              PRINT USING "  &          &  "; Master.Namae;
                                                                  '商品名
                              PRINT USING "#####"; Torihiki.Suuryou;     '数量
```

```
                    PRINT USING "#####,###"; Torihiki.Tanka;       '単価
                    PRINT USING "####,###,###"; Torihiki.Kingaku;
                                                                   '金額
                    Zandaka = Zandaka + Torihiki.Kingaku   '残高の計算
                    PRINT USING "####,###,###"; Zandaka     '残高の表示
                CASE "2"                    '入金取引の場合
                    PRINT USING "####"; K;                  'レコード番号
                    PRINT USING "  &       &  "; Torihiki.Hizuke;'日付
                    PRINT USING "#####"; Torihiki.Syoubangou;
                                                            '入金区分番号
                    SELECT CASE Torihiki.Syoubangou
                        CASE 1: kubun$ = "現金小切手"
                        CASE 2: kubun$ = "振込"
                        CASE 3: kubun$ = "手形"
                    END SELECT
                    PRINT USING "    入金区分:&          &"; kubun$;
                                                                '入金区分
                    PRINT USING "####,###,###"; orihiki.Kingaku;
                                                                  '入金額
                    Zandaka = Zandaka - Torihiki.Kingaku   '残高の計算
                    PRINT "            ";
                    PRINT USING "####,###,###"; Zandaka     '残高の表示
                CASE "3"                    '前月繰越の場合
                    PRINT USING "####"; K;                  'レコード番号
                    PRINT USING "  &       &  "; Torihiki.Hizuke;'日付
                    PRINT "          前月繰越                   ";
                    PRINT USING "####,###,###"; Torihiki.Kingaku;
                                                                   '金額
                    Zandaka = Zandaka + Torihiki.Kingaku   '残高の計算
                    PRINT USING "####,###,###"; Zandaka     '残高の表示
            END SELECT
            L = L + 1
        END IF
        IF L MOD 24 THEN
            PRINT "何かキーを押すと続けます。";
            DO
            LOOP UNTIL INKEY$ <> ""
            LOCATE CSRLIN, 1
        END IF
    NEXT K
    PRINT          "--------------------------------------------------------
                    <前行続き>  ------------------"
    PRINT "何かキーを押してください。"
    DO
    LOOP UNTIL INKEY$ <> ""
  END IF
NEXT I
CLOSE #1, #2
END
```

[請求書の印刷のサブプロシージャ]

```
DEFINT A-Z
'売掛管理 5        *****請求書の印刷*****
TYPE Mastertype                              'マスターファイルのレコードの型
    Namae       AS STRING * 10               '得意先・商品の名前は全角で5字
    S           AS INTEGER                   '削除のシルシ
END TYPE
TYPE Torihikitype
    kubun       AS STRING * 1
    Hizuke      AS STRING * 8
    TokuiBangou AS INTEGER
    Syoubangou  AS INTEGER
    Suuryou     AS INTEGER
    Tanka       AS LONG
    Kingaku     AS LONG
    S           AS INTEGER
END TYPE
DIM Master      AS Mastertype
DIM Torihiki    AS Torihikitype
DIM Zandaka     AS LONG
CLS
PRINT "     *****請求書の印刷*****"
PRINT
VIEW PRINT 4 TO 25
OPEN "B:Master.dat" FOR RANDOM AS #1 LEN = LEN(Master)
OPEN "B:Torihiki.dat" FOR RANDOM AS #2 LEN = LEN(Torihiki)
FOR I = 1 TO 50
    GET #1, I, Master
    IF Master.S THEN
        LPRINT CHR$(12)                      '改ページ
        LPRINT : LPRINT
        LPRINT "                                                   ";
            <前行続き>"     "; DATE$
        LPRINT "                              請  求  書"
        LPRINT : LPRINT USING "   NO. ###"; I
        LPRINT USING "  お得意様  &          &  殿"; Master.Namae
        LPRINT "                                              ";
            <前行続き> "(株) QB電器販売"
        LPRINT     "-------------------------------------------------
            <前行続き>  -------------------"
        LPRINT " ﾚｺｰﾄﾞ  日付  商品番号  商品名        数量    単  価";
            <前行続き>"    金     額     残     高"
        LPRINT      "------------------------------------------------
            <前行続き>  -------------------"
        J = LOF(2) ¥ LEN(Torihiki)
        Zandaka = 0
        FOR K = 1 TO J
            GET #2, K, Torihiki
            IF Torihiki.S AND I = Torihiki.TokuiBangou THEN
                SELECT CASE Torihiki.kubun
                    CASE "1"
                        LPRINT USING "####"; K;
```

```
                    LPRINT USING "  &      &  "; Torihiki.Hizuke;
                    LPRINT USING "#####"; Torihiki.Syoubangou;
                    GET #1, Torihiki.Syoubangou, Master
                    LPRINT USING "  &        &  "; Master.Namae;
                    LPRINT USING "#####"; Torihiki.Suuryou;
                    LPRINT USING "#####,###"; Torihiki.Tanka;
                    LPRINT USING "####,###,###"; Torihiki.Kingaku;
                    Zandaka = Zandaka + Torihiki.Kingaku
                    LPRINT USING "####,###,###"; Zandaka
                CASE "2"
                    LPRINT USING "####"; K;
                    LPRINT USING "  &      &  "; Torihiki.Hizuke;
                    LPRINT USING "#####"; Torihiki.Syoubangou;
                    SELECT CASE Torihiki.Syoubangou
                        CASE 1: kubun$ = "現金小切手"
                        CASE 2: kubun$ = "振込"
                        CASE 3: kubun$ = "手形"
                    END SELECT
                    LPRINT USING "   入金区分:&        &"; kubun$;
                    LPRINT USING "####,###,###"; Torihiki.Kingaku;
                    Zandaka = Zandaka - Torihiki.Kingaku
                    LPRINT "         ";
                    LPRINT USING "####,###,###"; Zandaka
                CASE "3"
                    LPRINT USING "####"; K;
                    LPRINT USING "  &      &  "; Torihiki.Hizuke;
                    LPRINT "   前月繰越                          ";
                    LPRINT USING "####,###,###"; Torihiki.Kingaku;
                    Zandaka = Zandaka + Torihiki.Kingaku
                    LPRINT USING "####,###,###"; Zandaka
            END SELECT
        END IF
    NEXT K
    LPRINT        "------------------------------------------------------
            <前行続き>   ------------------"
    LPRINT USING "                       御請求額合計 ####,###,###";Zandaka
    LPRINT        "------------------------------------------------------
            <前行続き>   ------------------"
    PRINT "何かキーを押してください。"
    DO
    LOOP UNTIL INKEY$ <> ""
  END IF
NEXT I
CLOSE #1, #2
END
```

7．6　売掛管理のまとめ

いよいよ、売掛管理のまとめの段階です。まとめる要領は次のとおりです。

1． 売掛管理の機能を画面のメニューから選択するメインプロシージャを作る。

2． これまでのメインプロシージャをサブプロシージャとして、結合する方法でそれぞれのサブプロシージャを複写する。

3． レコード型の設定は、メインプロシージャで行うのでサブプロシージャから削除する。

4． インクルードファイルの設定も、メインプロシージャで行うのでサブプロシージャから削除する。

5． 結合の際サブプロシージャも同時に結合されるので、これを独立したサブプロシージャとして設定する。

6． Helpsub:というサブルーチンはメインにおく必要があるので、取引の入力のサブプロシージャから分離して、メインに移動する。

7． プログラムの中でENDで強制終了する場合は、EXIT SUBとしてサブプロシージャから抜けるようにする。

売掛管理のプログラムの概要は次のとおりです。
（前節に同じ部分が多いので、一部省略します。）

```
DEFINT A-Z
DECLARE SUB Urikake (I, K$, S$)
DECLARE SUB Nyuukin (I, K$, S$)
DECLARE SUB Kurikosi (I, K$, S$)
DECLARE SUB Nyuuryoku (Y, X, S$)
DECLARE SUB MasterInput ()
DECLARE SUB TorihikiInput ()
DECLARE SUB ItiranTeiseiSakuzyo ()
DECLARE SUB TokuisakiItiran ()
DECLARE SUB Seikyuusyo ()
DECLARE SUB Teisei ()
DECLARE SUB Sakuzyo ()
'売掛管理        *****売掛管理のまとめ*****
'$INCLUDE: 'QB.BI'                          'インクルードファイルの設定
'$INCLUDE: 'GEN.BI'
```

```
TYPE Mastertype                                 'マスターファイルのレコードの型
     Namae          AS STRING * 10              '得意先・商品の名前は全角で5字
     S              AS INTEGER                  '削除のシルシ
END TYPE
TYPE Torihikitype
     Kubun          AS STRING * 1
     Hizuke         AS STRING * 8
     TokuiBangou    AS INTEGER
     SyouBangou     AS INTEGER
     Suuryou        AS INTEGER
     Tanka          AS LONG
     Kingaku        AS LONG
     S              AS INTEGER
END TYPE
DIM Master          AS Mastertype               'マスターレコードの設定
DIM Torihiki        AS Torihikitype             '取引レコードの設定
DO
     VIEW PRINT 1 TO 25
     CLS
     LOCATE 1, 20: COLOR 2:PRINT "****************************************"
     LOCATE 2, 20:         PRINT "****************************************"
     LOCATE 3, 20:         PRINT "*********   売 掛 管 理      *********"
     LOCATE 4, 20:         PRINT "****************************************"
     LOCATE 5, 20:         PRINT "****************************************"
     PRINT
     LOCATE 8, 20: COLOR 6:PRINT "1: マスターファイルの入力・削除"
     LOCATE 10, 20:        PRINT "2: 取引ファイルの入力"
     LOCATE 12, 20:        PRINT "3: 取引ファイルの一覧、訂正、削除"
     LOCATE 14, 20:        PRINT "4: 得意先別売掛取引の一覧"
     LOCATE 16, 20:        PRINT "5: 請求書の印刷"
     LOCATE 18, 20:        PRINT "6: 終了"
     LOCATE 20, 22:COLOR 7:PRINT"機能を選択して番号を入力してください。-->"
     DO
          Y$ = INKEY$
     LOOP UNTIL Y$ >= "1" AND Y$ <= "6"
     SELECT CASE Y$
          CASE "1": MasterInput                 '1ならばマスターファイルの入力
          CASE "2": TorihikiInput               '2ならば取引ファイルの入力
          CASE "3": ItiranTeiseiSakuzyo         '3ならば一覧表示、訂正、削除
          CASE "4": TokuisakiItiran             '4ならば得意先別の一覧表示
          CASE "5": Seikyuusyo                  '5ならば請求書の印刷
          CASE "6": END                         '6ならば終了
     END SELECT
LOOP
END
Helpsub:                                        'ヘルプ画面表示のサブルーチン
     HX = POS(0): HY = CSRLIN                   'カーソルの現在位置を得る
        (ヘルプ画面を表示するサブルーチン、一部省略)
RETURN
```

［一覧、訂正、削除のサブプロシージャ］

```
SUB ItiranTeiseiSakuzyo
'売掛管理３　　　　*****売掛・入金ファイルの画面表示、訂正、削除*****
DIM Master      AS Mastertype               'マスターレコードの設定
                 （一部省略、売掛管理３に同じ）
CLOSE #1, #2                                'ファイルを閉じる
END SUB
```

［繰越の入力のサブプロシージャ］

```
SUB Kurikosi (I, K$, S$)                    'サブプロシージャ　繰越の入力
DIM Master      AS Mastertype               'マスターレコードの設定
                 （一部省略）
LOOP
END SUB
```

［マスターファイルの入力のサブプロシージャ］

```
SUB MasterInput
DEFINT A-Z
'売掛管理１　　　　*****得意先・商品マスターファイルの登録・削除*****
DIM Master      AS Mastertype               'マスターファイルのレコード設定
                 （一部省略）
CLOSE #1                                    'ファイルを閉じる
END SUB
```

［入金のサブプロシージャ］

```
SUB Nyuukin (I, K$, S$)                     'サブプロシージャ　入金
DIM Master      AS Mastertype               'マスターレコードの設定
                 （一部省略）
LOOP
END SUB
```

［入力のためのサブプロシージャ］

```
SUB Nyuuryoku (Y, X, S$E)
S$ = ""                                     '入力された文字を連結する文字
LOCATE Y, X                                 '指示された位置へカーソル移動
                 （一部省略）
LOOP
END SUB
```

［削除のサブプロシージャ］

```
SUB Sakuzyo
DIM Torihiki    AS Torihikitype
                 （一部省略）
END SUB
```

[請求書の印刷のサブプロシージャ]

```
SUB Seikyuusyo
'売掛管理５        *****請求書の印刷*****
DIM Master        AS Mastertype
                   (一部省略)
NEXT I
END SUB
```

[訂正のサブプロシージャ]

```
SUB Teisei
DIM Master        AS Mastertype
                   (一部省略)
PUT #2, R, Torihiki                          '訂正済みのレコードの書き込み
END SUB
```

[得意先別一覧表示サブプロシージャ]

```
SUB TokuisakiItiran                          'サブプロシージャ得意先別一覧表示
'売掛管理４        *****得意先別売掛入金取引の画面表示*****
DIM Master        AS Mastertype              'マスターレコードの設定
                   (一部省略)
CLOSE #1, #2
END SUB
```

[取引ファイルの入のサブプロシージャ]

```
SUB TorihikiInput
'売掛管理２        *****売掛・入金ファイルの入力*****
DIM Master        AS Mastertype              'マスターのレコードの型
                   (一部省略)
KEY(31) GOSUB Helpsub:                       'HELPを押したらヘルプ画面を表示
                          (一部省略)
LOOP
END SUB
```

[売掛取引の入力のサブプロシージャ]

```
SUB Urikake (I, K$, S$)
DIM Master        AS Mastertype              'マスターレコードの型
                   (一部省略)
LOOP
END SUB
```

第❽章

お絵描きソフト

マウスの使い方の基本を学びます。

８．１　マウスで点を描く

マウスを使うためには、Ｑｕｉｃｋ　Ｂａｓｉｃのマウスに関するライブラリを利用する必要があります。それは、次のようなファイルです。

ライブラリ名　MOUSE.LIB
クイックライブラリ名　MOUSE.QLB
インクルードファイル名　QB.BI　MOUSE.BI

これらのファイルはすでにインストールされているので、直ちにマウスを使用できます。

これまでＱＢを起動する際には、漢字変換をするために次のように入力していました。

```
A>QB /LGEN
```

マウスを使うためには、次のように起動します。

```
A>QB /LMOUSE
```

さらに、プログラムの先頭でインクルードファイルを次のように指定します。

```
'$INCLUDE: 'QB.BI'
'$INCLUDE: 'MOUSE.BI'
```

これで、準備は終了しました。

点を描くプログラムは、次のとおりです。

```
DEFINT A-Z
'マウス１            ****点を描く****
'$INCLUDE: 'QB.BI'                          'インクルードファイルの設定
'$INCLUDE: 'MOUSE.BI'
CLS
MouseInit                                   'マウスの初期化
MouseShow                                   'マウスの表示
DO
    DO
        MousePush 0, Y, X, Hidari, Migi
                                'マウスの左ボタンが押された時点の位置を返す
    LOOP UNTIL Hidari = -1                  '左ボタンが押されるまで繰り返す
    PSET (X, Y), 2                          '左ボタンが押された位置へ点を描く
LOOP                                        '繰り返す
END
```

マウスを使うためには、まず、MouseInitでマウスを初期化します。

次に、MouseShowでマウスを表示します。特に位置を指定してしないので画面中央に表示されます。

MousePushはマウスのボタンが押された位置の座標を得るサブプロシージャです。

MousePushの機能は次のとおりです。

MousePush 0, Y, X, Hidari, Migi

- Migi：右ボタンを押すと-1、押さないと0を返します。
- Hidari：左ボタンを押すと-1、押さないと0を返します。
- X：マウスの横方向の位置（0から639まで）を返します。
- Y：マウスの縦方向の位置（0から399まで）を返します。
- 0：0とすると左ボタンを押したときのマウスの位置が返されます。1とすると右ボタンを押したときのマウスの位置が返されます。

MousePushの機能を理解するために、次のプログラムを実行してください。

```
DEFINT A-Z
'マウス2                ****MousePushの機能****
'$INCLUDE: 'QB.BI'
'$INCLUDE: 'MOUSE.BI'
CLS
MouseInit                                'マウスの初期化
MouseShow                                'マウスの表示
DO
    MousePush 0, Y, X, Hidari, Migi      'マウスの左ボタンが押された位置を取得
    PRINT Y, X, Hidari, Migi
LOOP                                     '繰り返す
```

ここで、MousePush 0, Y, X, Hidari, MigiをMousePush 1, Y, X, Hidari, Migiと取り替えて実行してください。

MousePush 0, X, Y, Hidari, Migi では、左ボタンを押した時のマウスの位置（X、Y）が返され、MousePush 1, X, Y, Hidari, Migi では、右ボタンを押した時のマウスの位置（X、Y）が返されていることが分ります。

また、押されていない状態でもそれぞれのデータが変化しないままで返されていることが分ります。

次に、連続して点を描くプログラムをつくります。

MousePush では、ボタンを押した一瞬のマウスの位置を返すだけなので、連続して点を描くことができません。そこで、MousePoll を使います。

```
DEFINT A-Z
'マウス3                ****連続の点を描く****
'$INCLUDE: 'QB.BI'
```

```
'$INCLUDE: 'MOUSE.BI'
CLS
MouseInit                                    'マウスの初期化
MouseShow                                    'マウスの表示
DO
    DO
        MousePoll Y, X, Hidari, Migi
                                   'マウスの左ボタンが押された時点の位置を返す
    LOOP UNTIL Hidari = -1                   '左ボタンが押されるまで繰り返す
    PSET (X, Y), 2                           '点を描く
LOOP
END
```

今度は、左ボタンを押している間に点が連続して描かれるようになりました。

MousePoll は、ボタンを押しても押さなくても、いつでも現在時点のマウスの位置を返してきます。

このことを、次のプログラムで確かめてください。

```
DEFINT A-Z
'マウス4              ****MousePollの機能****
'$INCLUDE: 'QB.BI'
'$INCLUDE: 'MOUSE.BI'
CLS
MouseInit                                    'マウスの初期化
MouseShow                                    'マウスの表示
DO
    MousePoll Y, X, Hidari, Migi             'マウスの現在位置を返す
    PRINT Y, X, Hidari, Migi
LOOP                                         '繰り返す
```

MousePoll の機能は、次のとおりです。

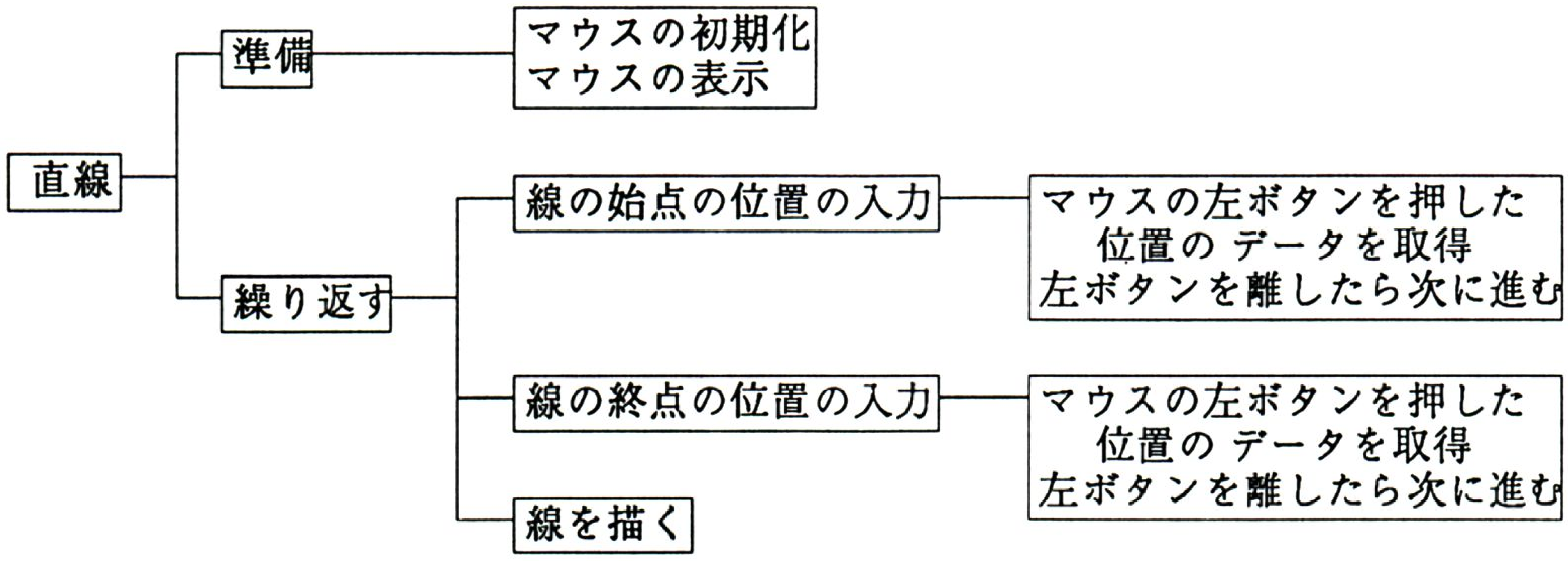
MousePoll Y, X, Hidari, Migi

- Migi：右ボタンを押すと-1、押さないと0を返します。
- Hidari：左ボタンを押すと-1、押さないと0を返します。
- X：マウスの横方向の現在位置（0から639まで）を返します。
- Y：マウスの縦方向の現在位置（0から399まで）を返します。

8．2　第二画面の利用

次に、直線を描くプログラムを作ります。

前節と同様に、MousePush及びMousePollを利用します。プログラムの構造は次のとおりです。




```
DEFINT A-Z
'マウス５                ****線を描く****          (MOUSE5.BAS)
'$INCLUDE: 'QB.BI'
'$INCLUDE: 'MOUSE.BI'
CLS
MouseInit                                          'マウスの初期化
MouseShow                                          'マウスの表示
DO
    LOCATE 23, 5: PRINT "線の始点を指示してください。"
    DO
        MousePush 0, Y1, X1, Hidari, Migi
                                                   '左ボタンを押した時点の位置を返す
    LOOP UNTIL Hidari = -1                         '左ボタンを押すまで繰り返す
    DO
        MousePoll Y, X, Hidari, Migi               '現在のマウスの位置等を取得
    LOOP UNTIL Hidari = 0                          '左ボタンを離してから次に進む
    LOCATE 23, 5: PRINT "線の終点を指示してください。"
    DO
        MousePush 0, Y2, X2, Hidari, Migi
                                                   '左ボタンを押した時点の位置を返す
    LOOP UNTIL Hidari                              '左ボタンを押すまで繰り返す
    LINE (X1, Y1)-(X2, Y2), 2                      '線を描く
    DO
        MousePoll Y, X, Hidari, Migi               '現在のマウスの位置等を取得
    LOOP UNTIL NOT Hidari                          '左ボタンを離してから次に進む
LOOP                                               '繰り返す
END
```

MousePushによって、左ボタンを押した時点のマウスの位置を取得し、MousePollによって、左ボタンを離した時点の位置を取得して、次へ進みます。

もしも、マウスの左ボタンを離す部分を省略すると、マウスの左ボタンを一度押すことによって線の始点と終点の位置の取得が一挙に終ってしまい、始点と終点が同じ位置になります。なぜなら、人間の指の動きは、パソコンに比してあまりに遅く、終点の判定の前に左ボタンから指を離すことができないからです。

このプログラムには終了の判定がないので、ストップキーを押して止めてください。（いずれ、終了するルーチンを加える予定です。）

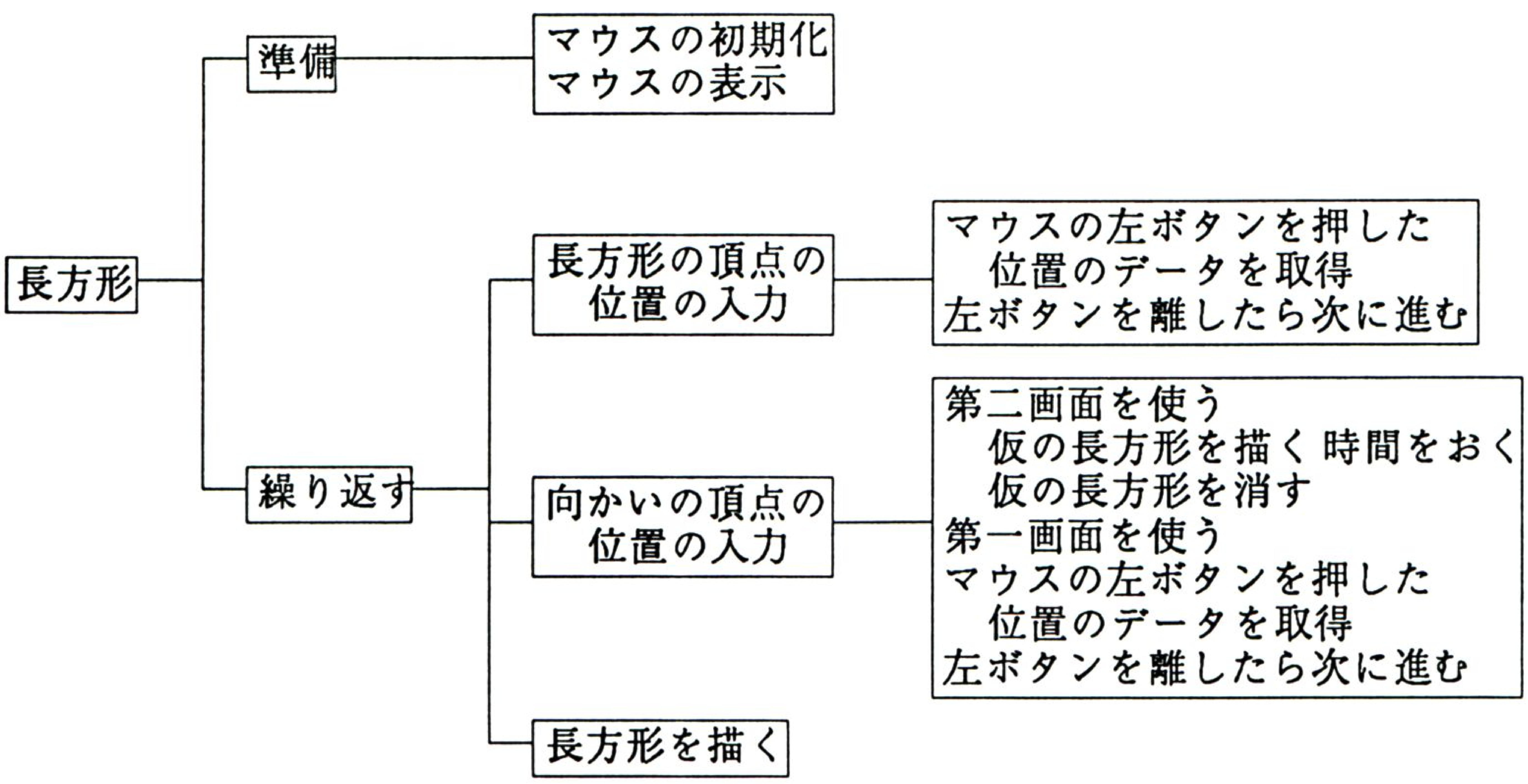
さて、このプログラムを使ってみていかがでしょうか。線を引く際に仮の線が表示されないので、どのように描かれるかわからず、非常に使いづらいと思います。そこで、仮の線を描く工夫をします。

また、画面をよく見るとマウスのために線の一部が欠けていますから、これも改善する必要があります。

ＰＣ９８０１シリーズは、グラフィック画面を２画面もっています。（一部に例外あり）そこで、この第二画面を利用して仮の線を引きます。しかし、裏の第二画面のみを見ていたのでは、すでに描いた絵との関連がとれないので、第一画面と第二画面とを交互に見るようにします。

また、仮の線は、表示した後少しの時間をとってすぐに消去します。

上記の部分を改善して長方形を描くプログラムを作りました。プログラムの構造は次のとおりです。



```
DEFINT A-Z
'マウス6              ****長方形を描く　その1****          (MOUSE6.BAS)
'$INCLUDE: 'QB.BI'
'$INCLUDE: 'MOUSE.BI'
CLS
MouseInit                                'マウスの初期化
MouseMode 1                              'ソフトウェアカーソルにする
MouseShow                                'マウスの表示
VIEW PRINT 23 TO 25                      'テキスト画面の範囲を設定
DO
    CLS 2                                'テキスト画面の画面消去
    LOCATE 23, 5: PRINT "長方形の頂点を指示してください。"
    DO
        MousePush 0, Y1, X1, Hidari, Migi
                                         '左ボタンを押した時点の位置を返す
    LOOP UNTIL Hidari                    '左ボタンを押すまで繰り返す
    DO
        MousePoll Y, X, Hidari, Migi     '現在のマウスの位置等を取得
    LOOP UNTIL NOT Hidari                '左ボタンを離してから次に進む
    CLS 2                                'テキスト画面の画面消去
    LOCATE 23, 5: PRINT "長方形の向いの頂点を指示してください。"
    DO
        MousePoll Y3, X3, H3, M3         '現在のマウスの位置等を取得
        SCREEN 0, , 1, 1                 '第二画面を使う
        LINE (X1, Y1)-(X3, Y3), 2, B     '長方形を描く
        FOR I = 1 TO 1000: NEXT          '時間をおく
        LINE (X1, Y1)-(X3, Y3), 0, B     '長方形を消す
        SCREEN 0, , 0, 0                 '第一画面を使う
        MousePush 0, Y2, X2, Hidari, Migi
                                         '左ボタンを押した時点の位置を返す
    LOOP UNTIL Hidari                    '左ボタンを押すまで繰り返す
    LINE (X1, Y1)-(X2, Y2), 2, B         '長方形を描く
    DO
        MousePoll Y, X, Hidari, Migi     '現在のマウスの位置等を取得
    LOOP UNTIL NOT Hidari                '左ボタンを離してから次に進む
LOOP                                     '繰り返す
END
```

```
MouseMode 1
```

ソフトウェアカーソルを使用できます。カーソルの形が長方形なので精度の高い指定はできないのですが、グラフィック画面に傷をつけることがないのでこれを使いました。

```
SCREEN 0, , 0, 0
```

第一画面を使用するステートメントです。SCREEN 0, , 1, 1なら、第二画面です。SCREEN の書式は次のとおりです。

```
SCREEN 0, 0, 0, 0
       │  │  │  └▷ 表示するページ（画面）の指定　第一画面：0　第二画面：1
       │  │  └───▷ 書き込むページ（画面）の指定　第一画面：0　第二画面：1
       │  └▷ パレットスイッチ
       └▷ スクリーンモード
```

```
LINE (X1, Y1)-(X3, Y3), 0, B
```

第二画面の仮の長方形を消すために、CLSを使わず黒色の長方形を描いています。

```
FOR I = 1 TO 1000: NEXT
```

仮の長方形を表示しておく時間を保持するために、1000回数えたのです。あまりに速いので、とても1000回数えたとは思えないくらいです。

さて、上記のプログラムでは、左ボタンを離したことを確認しながら進めていますが、今度は、線の終点をマウスの左ボタンを離した時点の位置に訂正してみました。この場合の操作法は、次のようになります。

1. 左ボタンを押して始点を指示する。
2. 左ボタンを押したままでマウスを移動し、離して終点の位置を指示する。

一部修正したプログラムは、次のとおりです。

```
DEFINT A-Z
'マウス7              ****長方形を描く　その2****              (MOUSE7.BAS)
'$INCLUDE: 'QB.BI'
'$INCLUDE: 'MOUSE.BI'
CLS
MouseInit                                  'マウスの初期化
MouseMode 1                                'ソフトウェアカーソルにする
MouseShow                                  'マウスの表示
VIEW PRINT 23 TO 25                        'テキスト画面の範囲を設定
```

```
DO
    CLS 2                                       'テキスト画面の画面消去
    LOCATE 23, 5: PRINT "長方形の頂点を指示してください。"
    DO
        MousePush 0, Y1, X1, Hidari, Migi
                                                '左ボタンを押した時点の位置を返す
    LOOP UNTIL Hidari                           '左ボタンを押すまで繰り返す
    CLS 2                                       'テキスト画面の画面消去
    LOCATE 23, 5: PRINT "長方形の向いの頂点を指示してください。"
    DO
        MousePoll Y3, X3, H3, M3                '現在のマウスの位置等を取得
        SCREEN 0, , 1, 1                        '第二画面を使う
        LINE (X1, Y1)-(X3, Y3), 2, B            '長方形を描く
        FOR I = 1 TO 1000: NEXT                 '時間をおく
        LINE (X1, Y1)-(X3, Y3), 0, B            '長方形を消す
        SCREEN 0, , 0, 0                        '第一画面を使う
        MouseRelease 0, Y2, X2, Hidari, Migi
                                                '左ボタンを押した時点の位置を返す
    LOOP UNTIL NOT Hidari                       '左ボタンを押すまで繰り返す
    LINE (X1, Y1)-(X2, Y2), 2, B                '長方形を描く
LOOP                                            '繰り返す
END
```

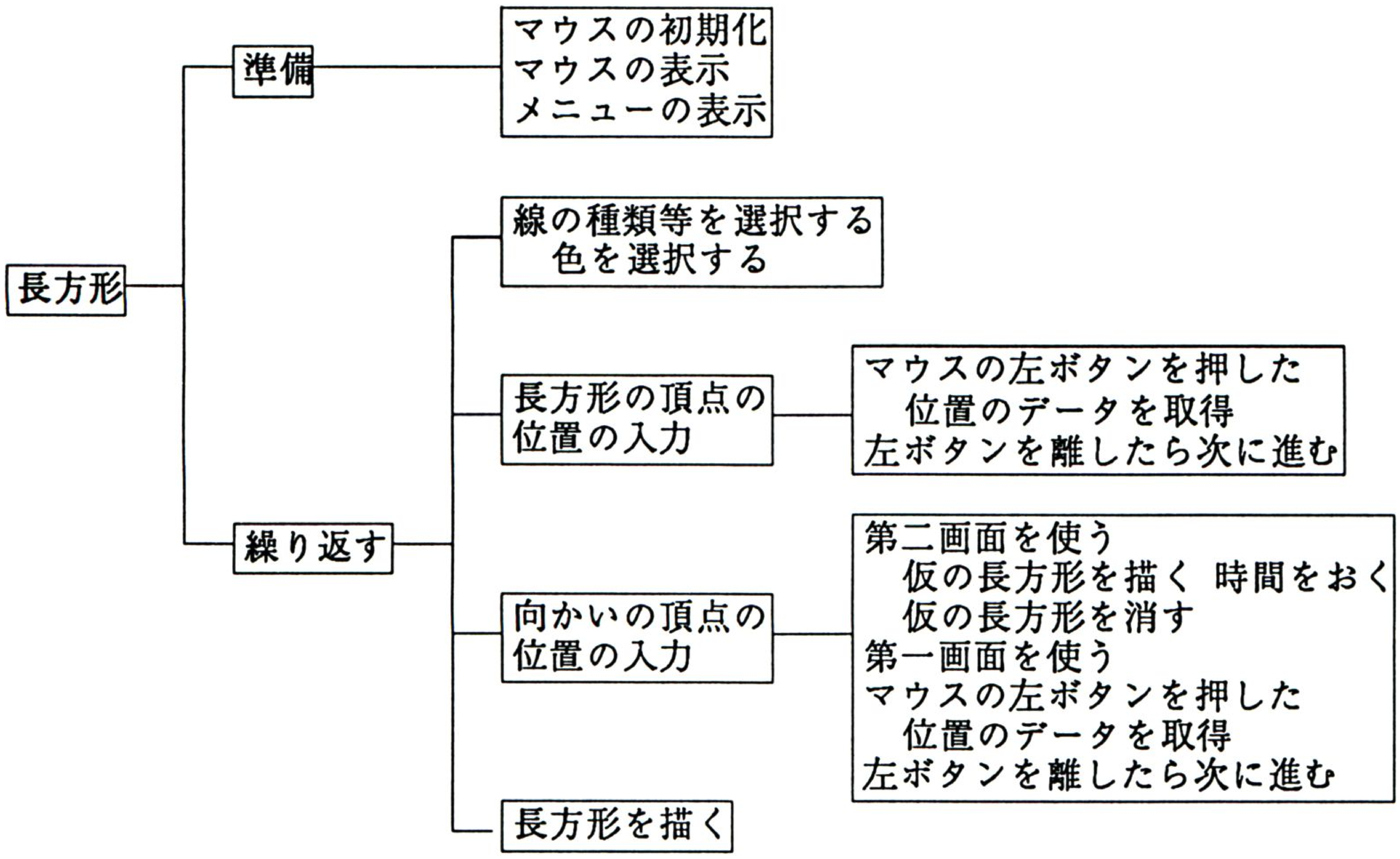
```
MouseRelease 0, Y2, X2, Hidari, Migi
```

左ボタンを離れた時点のマウスの位置等を取得します。

8．3　マウスによるメニュー選択

線の種類や色を指定して長方形を描くプログラムを作ります。

線の種類や色の指定は、他の図形を描くプログラムでも使いますから、サブプロシージャにします。プログラムの構造は、次のとおりです。




```
DEFINT A-Z
DECLARE SUB Sensyu (S)
DECLARE SUB Iro (C)
DECLARE SUB Menu ()
'マウス８                ****メニュー選択　長方形****
'$INCLUDE: 'QB.BI'                          'インクルードファイルの設定
'$INCLUDE: 'MOUSE.BI'
CLS
MouseInit                                   'マウスの初期化
MouseMode 1                                 'ソフトウェアカーソルにする
MouseShow                                   'マウスの表示
Menu                                        'メニューを表示
DO
    Sensyu S                                '線の種類を選択する
    Iro C                                   '色を選択する
    CLS 2                                   'テキスト画面を消去
    LOCATE 23, 5: PRINT "長方形の頂点を指示してください。"
    DO
        MousePush 0, Y1, X1, Hidari, Migi   '左ボタンを押した時点の位置を返す
    LOOP UNTIL Hidari                       '左ボタンを押すまで繰り返す
```

```
    DO
        MousePoll Y, X, Hidari, Migi              '現在のマウスの位置等を取得
    LOOP UNTIL NOT Hidari                         '左ボタンを離してから次に進む
    CLS 2
    LOCATE 23, 5: PRINT "長方形の向いの頂点を指示してください。"
    DO
        MousePoll Y3, X3, H3, M3                  '現在のマウスの位置等を取得
        SCREEN 0, , 1, 1                          '第二画面を使う
        LINE (X1, Y1)-(X3, Y3), C, B              '長方形を描く
        FOR I = 1 TO 1000: NEXT                   '時間をおく
        LINE (X1, Y1)-(X3, Y3), 0, B              '長方形を描く
        SCREEN 0, , 0, 0                          '第一画面を使う
        MousePush 0, Y2, X2, Hidari, Migi         '左ボタンを押した時点の位置を返す
    LOOP UNTIL Hidari                             '左ボタンを押すまで繰り返す
    SELECT CASE S
        CASE 0
            LINE (X1, Y1)-(X2, Y2), C, B                           '実線で描く
        CASE 1
            LINE (X1, Y1)-(X2, Y2), C, B, &HF0F0                   '点線で描く
        CASE 2
            LINE (X1, Y1)-(X2, Y2), C, B, &HFF99                   '破線で描く
        CASE 3
            LINE (X1, Y1)-(X2, Y2), C, BF                    '塗りつぶして描く
    END SELECT
    DO
        MousePoll Y, X, Hidari, Migi              '現在のマウスの位置等取得
    LOOP UNTIL NOT Hidari                         '左ボタンを離してから次に進む
LOOP                                              '繰り返す
END
```

［メニューを画面表示するサブプロシージャ］

```
SUB Menu
VIEW PRINT 23 TO 25                 'テキスト画面の２３行目から２５行目まで使用する
LINE (100, 390)-(150, 390), 7                     '実線メニューの表示
LINE (150, 390)-(200, 390), 7, , &HF0F0           '点線メニューの表示
LINE (200, 390)-(250, 390), 7, , &HFF99           '破線メニューの表示
LINE (250, 385)-(300, 395), 7, BF                 '塗りつぶしメニューの表示
FOR I = 0 TO 7
    LINE (320 + (I * 30), 385)-(350 + (I * 30), 395), I, BF     '色メニューの表示
NEXT I
LINE (320, 385)-(530, 395), 7, B                  '色メニューの枠の表示
END SUB                                           'サブプロシージャを出る
```

[線の種類を選択するサブプロシージャ]

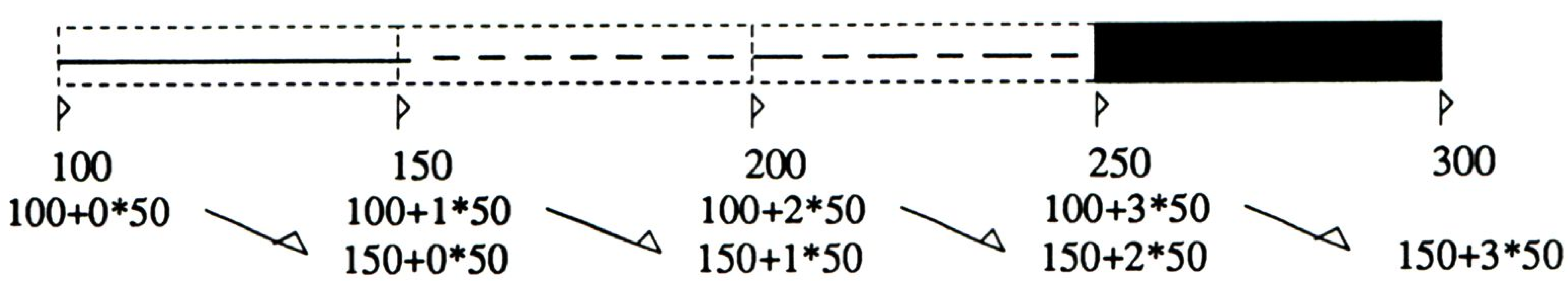
```
SUB Sensyu (S)
CLS 2                                        'テキスト画面（文字の画面）の消去
LOCATE 23, 5: PRINT "線種等を選択してください。"
MouseBorder 385, 100, 399, 300              'マウスの移動範囲をメニュー上に設定
DO                                           '繰り返す
    MousePush 0, Y, X, Hidari, Migi          '左ボタンを押した時点の位置を取得
LOOP UNTIL Hidari                            '左ボタンが押されるまで繰り返す
FOR S = 0 TO 3                               '線の番号の0から3まで繰り返す
    IF S > 100 + S * 50 AND S < 150 + S * 50 THEN        '範囲内ならば
        EXIT FOR                             'FOR・・・NEXTの外へ出る
    END IF
NEXT S
DO
    MousePoll Y, X, Hidari, Migi             '現在のマウスの位置等を取得
LOOP UNTIL NOT Hidari                        '左ボタンを離してから次に進む
MouseBorder 0, 0, 360, 639                   'マウスの移動範囲を描画画面上に設定
MouseLocate 200, 320                         'マウスを画面中央へ移動
END SUB                                      'サブプロシージャの外へ出る
```

```
IF X > 100 + S * 50 AND X < 150 + S * 50 THEN
```

線の種類の選択を判定する式です。各種類ごとに判定してもよいのですが、この式でプログラムを簡潔にします。

マウスが押された横方向の位置（X）が、100+線番号*50から150+線番号*50の範囲内にあることを利用して判定しています。

線の種類のメニューの画面上の横方向の位置(X)と判定のための式




```
EXIT FOR
```

線の種類を判定した結果は、Sとなってメインへ渡されます。

EXIT FORによって、NEXT Sの次へジャンプして、判定して得たSの値が変わらないようにします。

```
MouseBorder 385, 100, 399, 300
```

マウスが移動できる範囲の指定です。メニュー選択時には、マウスをメニューの範囲内におく必要がありますから、(385,100)から(399,300)の範囲を設定します。

MouseBorder 0, 0, 360, 639 は、マウスの移動範囲を、メニューの範囲から作画の範囲へ改めたものです。MouseLocate 200, 320 は、マウスを表示する位置の指定です。作画のために画面の中央に表示します。

メニューには、長方形の塗りつぶしの選択も加えています。

[色を選択するサブプロシージャ]

```
SUB Iro (C)
CLS 2                                             'テキスト画面（文字の画面）を消去
LOCATE 23, 5: PRINT "色を選択してください。"
MouseBorder 385, 320, 399, 560                    'マウスの移動範囲をメニュー上に設定
DO                                                '繰り返す
    MousePush 0, Y, X, Hidari, Migi               '左ボタンを押した時点の位置を取得
LOOP UNTIL Hidari                                 '左ボタンが押されるまで繰り返す
FOR C = 0 TO 7                                    '色番号0から7まで繰り返す
    IF X > 320 + C * 30 AND X < 350 + C * 30 THEN '範囲内ならば
        EXIT FOR                                  'FOR・・・NEXTの外へ出る
    END IF
NEXT C
DO
    MousePoll Y, X, Hidari, Migi
LOOP UNTIL NOT Hidari                             '左ボタンを離してから次に進む
MouseBorder 0, 0, 360, 639                        'マウスの移動範囲を描画画面上に設定
MouseLocate 200, 320                              'マウスを画面中央へ移動
END SUB                                           'サブプロシージャを出る
```

```
IF X > 320 + C * 30 AND X < 350 + C * 30 THEN
```

線の種類を選択する場合と同様に、マウスの位置（X）が色のメニューの中のいずれの位置にあるか判定します。図示すると次のとおりです。

黒	青	緑	水	赤	紫	黄	白
320+0*30 ～ 350+0*30	320+1*30 ～ 350+1*30	320+2*30 ～ 350+2*30	320+3*30 ～ 350+3*30	320+4*30 ～ 350+4*30	320+5*30 ～ 350+5*30	320+6*30 ～ 350+6*30	320+7*30 ～ 350+7*30

320　350　380　410　440　470　500　530　560

マウスに関するライブラリは、次のとおりです。

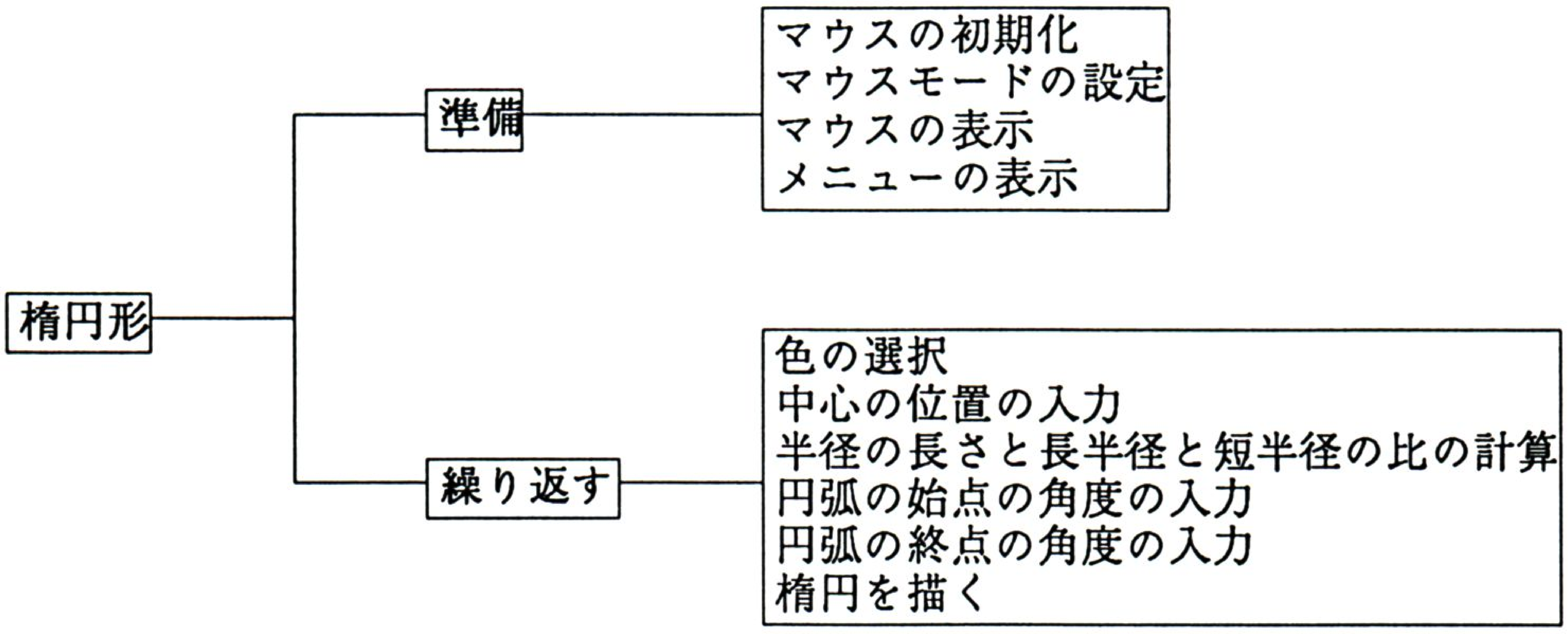
MouseInit	マウスの初期化
MouseShow	カーソルの表示
MouseMode	マウスのタイプの設定
MousePoll	マウスの現在の位置等の取得
MousePush	ボタンを押した時の位置等の取得
MouseRelease	ボタンを離した時の位置等の取得
MouseBorder	マウスの移動範囲の設定
MouseLocate	マウスの表示位置の設定
MouseHide	マウスの非表示
MouseReady%	マウスが利用可能の判定（関数）
SetMouseEvent	割り込み条件の設定

8．4　楕円と関数の利用

次に、楕円の作画をします。基本的には、長方形と同様のプログラムですが、円弧の始点と終点の角度を計算する必要があります。そこで、関数のサブプロシージャをつくって、円弧の角度の計算を行うことにします。

なお、円は楕円のルーチンを使うこととして、円を描くプログラムは、省略します。

プログラムの構造は、次のとおりです。




```
DEFINT A-Z
DECLARE FUNCTION Enko! (X1, Y1, XX, YY)
DECLARE SUB Sensyu (S)
DECLARE SUB Iro (C)
DECLARE SUB Menu ()
'マウス９           ****楕円****              (MOUSE9.BAS)
'$INCLUDE: 'QB.BI'                   '←インクルードファイルの設定
'$INCLUDE: 'MOUSE.BI'
CLS
 MouseInit                           '←マウスの初期化
 MouseMode 1                         '←ソフトウェアカーソルにする
 MouseShow                           '←マウスの表示
 Menu                                '←メニュー画面を表示
DO
     Iro C                           '←色の選択
     CLS 2                           '←テキスト画面の消去
     LOCATE 23, 5: PRINT "円の中心を指示してください。"
     DO
        MousePush 0, Y1, X1, Hidari, Migi
                                     '←左ボタンを押した時点の位置を返す
     LOOP UNTIL Hidari               '←左ボタンを押すまで繰り返す
     DO
        MousePoll Y, X, Hidari, Migi  '←マウスの現在の位置等を取得
     LOOP UNTIL NOT Hidari            '←左ボタンを離してから次に進む
```

```
    CLS 2                              '←テキスト画面の消去
    LOCATE 23, 5: PRINT "円の半径を指示してください。"
    DO
        MousePoll Y3, X3, H3, M3       '←現在のマウスの位置等を取得
        SCREEN 0, , 1, 1               '←第二画面の使用
        IF X3 <> X1 THEN               '↓仮の円を描く
            CIRCLE (X1, Y1), SQR((X1 - X3) ^ 2 + (Y1 - Y3) ^ 2),
                        <前行続き> C, , , ABS((Y1 - Y3) / (X1 - X3))
            FOR I = 1 TO 1000: NEXT    '←時間をおいて↓仮の円を消す
            CIRCLE (X1, Y1), SQR((X1 - X3) ^ 2 + (Y1 - Y3) ^ 2),
                        <前行続き> 0, , , ABS((Y1 - Y3) / (X1 - X3))
        END IF
        SCREEN 0, , 0, 0               '←第一画面の使用
        MousePush 0, Y2, X2, Hidari, Migi
                                       '←左ボタンを押した時点の位置を返す
    LOOP UNTIL Hidari                  '←左ボタンを押すまで繰り返す
    R = SQR((X1 - X2) ^ 2 + (Y1 - Y2) ^ 2)      '←半径の計算
    IF X1 <> X2 THEN                            '←分母が0でなければ
        Hi! = ABS((Y1 - Y2) / (X1 - X2)) '←長径と短径の比の計算
    ELSE
        Hi! = 1000                      '←分母が0ならば比は1000とする
    END IF
    DO
        MousePoll Y, X, Hidari, Migi  '←マウスの現在時点の位置等を取得
    LOOP UNTIL NOT Hidari              '←左ボタンを離してから次に進む
    CLS 2                              '←テキスト画面の消去
    LOCATE 23, 5: PRINT "    円弧の始点を指示してください。"
    DO
        MousePoll Y3, X3, H3, M3       '←マウスの現在の位置等を取得
    SCREEN 0, , 1, 1                   '←第二画面の使用
    IF X3 <> X1 THEN
        CIRCLE (X1, Y1), R, C, Enko!(X1, Y1, X3, Y3), , Hi!
                                       '←仮の円弧を描く
        FOR I = 1 TO 1000: NEXT        '←表示する時間をおく
        CIRCLE (X1, Y1), R, 0, Enko!(X1, Y1, X3, Y3), , Hi!
                                       '←仮の円弧を消す
    END IF
    SCREEN 0, , 0, 0                   '←第一画面の使用
    MousePush 0, Y2, X2, Hidari, Migi
                                       '←左ボタンを押した時点の位置を返す
LOOP UNTIL Hidari                      '←左ボタンを押すまで繰り返す
ST! = Enko!(X1, Y1, X2, Y2)            '←円弧の始点の計算
DO
    MousePoll Y, X, Hidari, Migi       '←マウスの現在の位置等を取得
LOOP UNTIL NOT Hidari                  '←左ボタンを離してから次に進む
CLS 2
LOCATE 23, 5: PRINT "    円弧の終点を指示してください。"
DO
    MousePoll Y3, X3, H3, M3           '←マウスの現在の位置等を取得
```

```
SCREEN 0, , 1, 1                                 '←第二画面を使う
    IF X3 <> X1 THEN
        CIRCLE (X1, Y1), R, C, ST!, Enko!(X1, Y1, X3, Y3), Hi!
                                                  '←仮の円弧を描く
        FOR I = 1 TO 1000: NEXT                   '←時間をおく
        CIRCLE (X1, Y1), R, 0, ST!, Enko!(X1, Y1, X3, Y3), Hi!
                                                  '←仮の円を消す
    END IF
    SCREEN 0, , 0, 0
    MousePush 0, Y2, X2, Hidari, Migi             '←左ボタンを押した時点の位置を返す
LOOP UNTIL Hidari                                 '←左ボタンを押すまで繰り返す
EN! = Enko!(X1, Y1, X2, Y2)                       '←円弧の終点の計算
DO
    MousePoll Y, X, Hidari, Migi                  '←マウスの現在の位置等を取得
LOOP UNTIL NOT Hidari                             '←左ボタンを離してから次に進む
IF ST! = EN! THEN                                 '←始点と終点が一致した場合は
    ST! = 0: EN! = 3.141593 * 1.9999999#          '←0から2πまでとして楕円を描く
END IF
CIRCLE (X1, Y1), R, C, ST!, EN!, Hi!              '←楕円・円弧を描く
DO
    MousePoll Y, X, Hidari, Migi                  '←マウスの現在の位置等を取得
LOOP UNTIL NOT Hidari                             '←左ボタンを離してから次に進む
LOOP                                              '←繰り返す
END
```

Enko!(X1, Y1, X3, Y3)は、円弧の始点と終点の角度を計算する関数を使っているところです。

角度の単位はラジアンですから、Enko! は浮動小数点型の関数です。全部の円弧を描く場合は、0から2πまでですが、これではCIRCLEが動かないようなので、0から1.9999999#πとして描くようにしてあります。

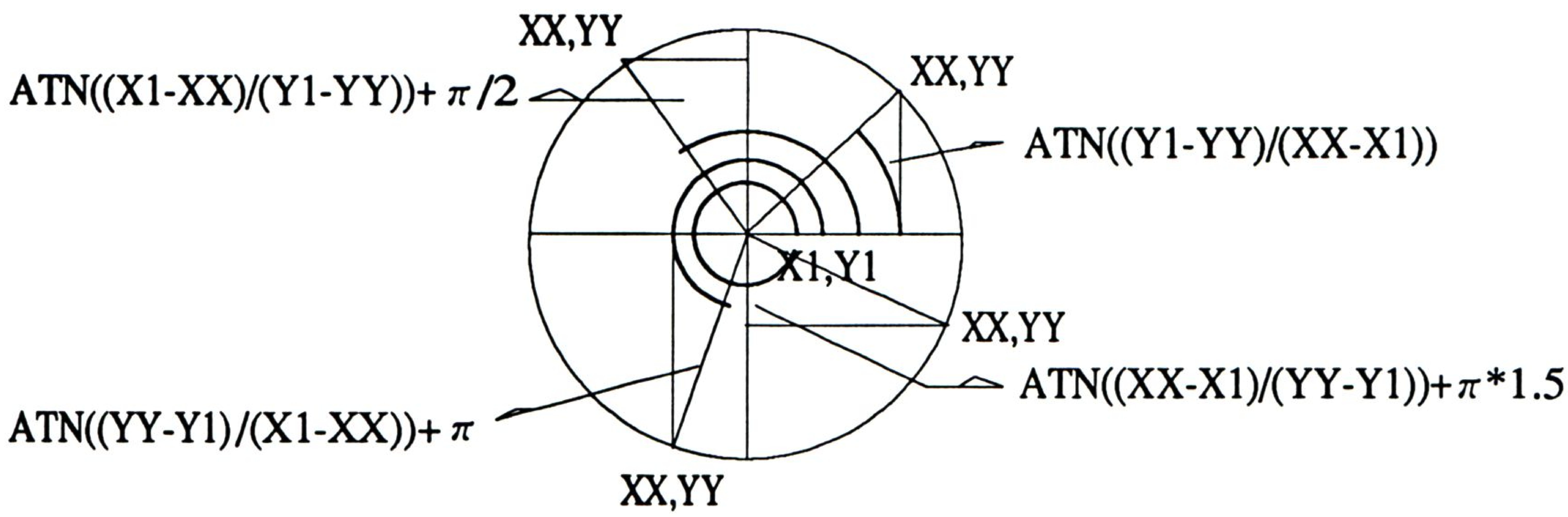
[円弧の始点と終点を計算するサブプロシージャ]

```
FUNCTION Enko! (X1, Y1, XX, YY)
IF XX <> X1 AND X1 < XX AND Y1 > YY THEN          '←第1象限の場合
    Enko! = ATN((Y1 - YY) / (XX - X1)): EXIT FUNCTION
END IF
IF YY <> Y1 AND X1 > XX AND Y1 > YY THEN          '←第2象限の場合
  Enko! = ATN((X1 - XX) / (Y1 - YY)) + (3.141593 / 2): EXIT FUNCTION
END IF
IF XX <> X1 AND X1 > XX AND Y1 < YY THEN          '←第3象限の場合
  Enko! = ATN((YY - Y1) / (X1 - XX)) + 3.141593: EXIT FUNCTION
END IF
IF YY <> Y1 AND X1 < XX AND Y1 < YY THEN          '←第4象限の場合
  Enko! = ATN((XX - X1) / (YY - Y1)) + 3.141593 * 1.5: EXIT FUNCTION
END IF
```

```
IF XX = X1 AND Y1 > YY THEN                         '←90度の場合
 Enko! = 3.141593 / 2: EXIT FUNCTION
END IF
IF XX = X1 AND Y1 < YY THEN                         '←270度の場合
 Enko! = 3.141593 * 1.5: EXIT FUNCTION
END IF
IF YY = Y1 AND X1 > XX THEN                         '←180度の場合
 Enko! = 3.141593: EXIT FUNCTION
END IF
IF YY = Y1 AND X1 < XX THEN                         '←0度の場合
    Enko! = 0: EXIT FUNCTION
END IF
END FUNCTION
```

円弧の始点と終点の角度の計算は、各象限ごとにタンジェント（TAN）の逆関数のATN（アークタンジェント）を使います。0,90,180,270度の場合は、計算式の分母が0になってATNでは計算でないので、別にしました。

次の図のように計算しています。



［色を選択するサブプロシージャ］

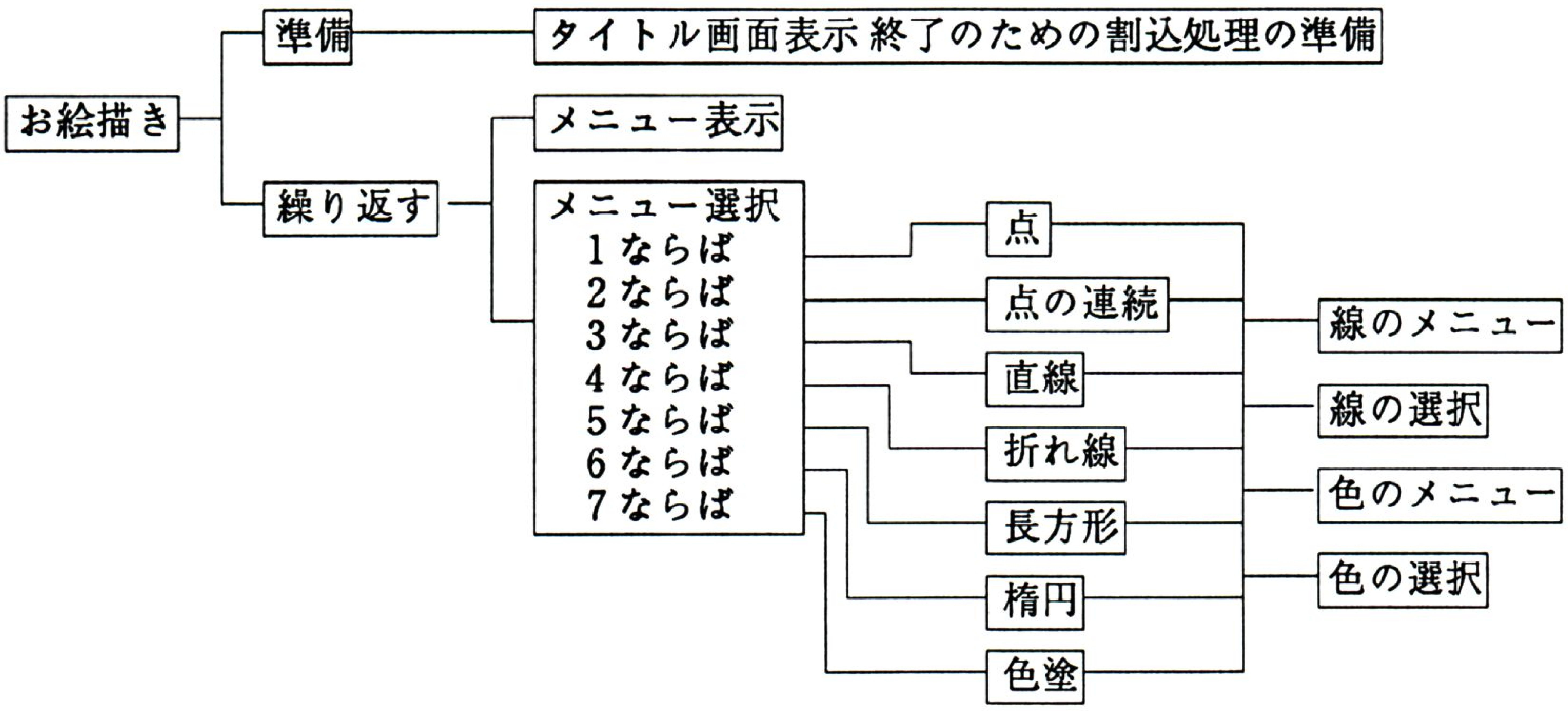
```
SUB Iro (C)
（前節と同じなので、省略します。）
END SUB
```

［色の選択のメニューを表示するサブプロシージャ］

```
SUB Menu
VIEW PRINT 23 TO 25
FOR I = 0 TO 7
 LINE (320 + (I * 30), 385)-(350 + (I * 30), 395), I, BF
NEXT I
LINE (320, 385)-(530, 395), 7, B
END SUB
```

8．5　お絵描きソフトのまとめ

いよいよ、お絵描きソフトにまとめます。
このプログラムの構造の概要は次のとおりです。



（1）メインプロシージャ

　左ボタンと右ボタンを同時に押せば終了できるようにしました。

　そのために、メインプロシージャの最後に終了のためのサブルーチンを用意し、さらに、このサブルーチンへ跳ぶための割込処理を設定しています。

```
DEFINT A-Z
DECLARE SUB SensyuMenu ()
DECLARE SUB Oresen ()
DECLARE SUB Daen ()
DECLARE SUB IroNuri ()
DECLARE SUB Tyouhoukei ()
DECLARE SUB TenRenzoku ()
DECLARE SUB Tyokusen ()
DECLARE SUB ZukeiMenu ()
DECLARE SUB Ten ()
DECLARE SUB IroMenu ()
DECLARE SUB ZukeiSentaku ()
DECLARE FUNCTION Enko! (X1, Y1, XX, YY)
DECLARE SUB Sensyu (S)
DECLARE SUB Iro (C)
'マウス１０          ****  お絵描き  ****   (MOUSE10.BAS)
'$INCLUDE: 'QB.BI'                   '←インクルードファイルの設定
'$INCLUDE: 'MOUSE.BI'
ON UEVENT GOSUB Syuuryou             '←終了のために割り込み処理をする設定
```

```
CLS                                      '←画面消去
COLOR 9: LOCATE 5, 20:  PRINT "**********************************"
COLOR 11: LOCATE 6, 20: PRINT "**********************************"
COLOR 10: LOCATE 7, 20: PRINT "**********************************"
COLOR 5: LOCATE 8, 20:  PRINT "**********************************"
COLOR 7: LOCATE 9, 20:  PRINT "         お絵かきソフト             "
COLOR 5: LOCATE 10, 20: PRINT "**********************************"
COLOR 14: LOCATE 11, 20:PRINT "**********************************"
COLOR 13: LOCATE 12, 20:PRINT "**********************************"
COLOR 12: LOCATE 13, 20:PRINT "**********************************"
COLOR 6: LOCATE 17, 20:PRINT "マウスの左ボタンを押すとスタートします。"
COLOR 5: LOCATE 19, 20:PRINT "マウスの右ボタンを押すと図形を選択できます。"
COLOR 4: LOCATE 21, 20:PRINT "マウスの左右のボタンを同時に押すと終了します。"
DO
 MousePoll Y, X, Hidari, Migi       '←現在時点のマウスの位置の取得
LOOP UNTIL Hidari                       '←左ボタンを押せば次へ
DO
 MousePoll Y, X, Hidari, Migi       '←現在時点のマウスの位置の取得
LOOP UNTIL NOT Hidari                   '←左ボタンを離せば次へ
CLS
 MouseInit                              '←マウスの初期化
 SetMouseEvent 8                        '←割り込み処理の条件の設定
 MouseMode 1                            '←マウスの種類の設定
 MouseShow                              '←マウスの表示
DO
  ZukeiMenu                            '←図形の選択メニューの表示
  ZukeiSentaku                           '←図形の選択と作画
LOOP
END
 Syuuryou:                           '←終了処理のためのサブルーチン
  MousePush 0, Y, X, Hidari, Migi
                                    '←左ボタンを押した時点の位置等を取得
  IF Hidari THEN                    '←左ボタンが押されているば
     CLS 2                                  '←画面消去して
     LINE (70, 383)-(560, 399), 0, BF       '←メニューを消去して
     MouseHide                              '←マウスを消して
     UEVENT OFF                             '←割り込み処理をOFF
     SetMouseEvent 0                '←割り込み処理の条件設定を消去
     END                            '←終了
  END IF
RETURN
```

ON UEVENT GOSUB Syuuryouは、終了のための割込処理を定義しています。Syuuryouは、メインプロシージャの後において、割込があったら、Syuuryouにある仕事をします。この場合は終了の条件が合えば、そのまま終了しますが、条件が合わない場合は、RETURNから、割り込みした位置へ戻ることになります。

SetMouseEvent 8 は、Mouseによる割込をするための条件を設定します。後ろの8が条件を表しています。

条件は次により設定できます。

ビット位置	4	3	2	1	0
	0	0	0	0	0

- ビット0：1ならば、カーソル位置の変化で割り込み
- ビット1：1ならば、左ボタンＯＮで割り込み
- ビット2：1ならば、左ボタンＯＦＦで割り込み
- ビット3：1ならば、右ボタンＯＮで割り込み
- ビット4：1ならば、右ボタンＯＦＦで割り込み

この場合は、右ボタンＯＮで割り込み処理を行うので、次のようにします。

ビット位置	4	3	2	1	0
	0	1	0	0	0

２進数の01000は、１０進数では8なので、SetMouseEvent 8 としたのです。

Ｑｕｉｃｋ　Ｂａｓｉｃではこのような割込処理をイベントトラッピングといっています。

ON UEVENT GOSUB ・・・・・は、ユーザー定義のイベントトラッピングで、使用者が定義して利用するものです。

イベントトラッピングには次のようなステートメントがあります。

ON COM(シリアルポート番号) GOSUB ・・・・通信で文字を受信したときの割り込み

ON KEY(番号) GOSUB ・・・・　キーを押したときの割り込み

番号・・1-10はファンクションキーの１－１０
11は上矢印キー　12は左矢印キー
13は右矢印キー　14は下矢印キー
30はSTOPキー　31はHELPキー
15-25はユーザー定義

ON PLAY(維持する音符の最小値) GOSUB ・・・・オルゴールを続けて鳴らすために音符を読み込むための割り込み

ON TIME(秒数) GOSUB ・・・・時間を経過したときの割り込み

なお、上記は、機能の設定のみで、実際に働かすには、COM(n) ON, KEY(n) ON, PLAY ON, TIMER ON を必要とします。

MouseHideは、マウスの表示を消します。

UEVENT OFFは、割込処理をＯＦＦにして、働かないようにします。割込処理のＯＮは、UEVENT ON で行い、これは図形を選択するサブプロシージャにあります。

SetMouseEvent 0 は、マウスの割り込みの条件設定を消去します。終了する前に

は、しっかりと後始末をとする必要があります

（２）図形選択のメニューを表示するサブプロシージャ

```
SUB ZukeiMenu          '←図形選択のためのメニュー表示
 CLS 2                              '←文字の画面を消去
 LINE (70, 383)-(560, 399), 0, BF          '←メニューを消去
 COLOR 7: LOCATE 23, 5: PRINT "図形を選択してください。"
 FOR I = 1 TO 7                        '←メニューの枠表示
  LINE (0 + (I * 70), 383)-(70 + (I * 70), 399), 7, B
 NEXT I
 FOR I = 1 TO 20
  X = RND * 70 + 70: Y = RND * 15 + 383     '←点のメニュー
  PSET (X, Y), 2
 NEXT I
 FOR R! = 0 TO 2 * 3.141593 STEP .1          '←点の連続のメニュー
  X! = 145 + R! * 10
  Y! = 391 + SIN(R!) * 6                 '←ＳＩＮカーブ
  PSET (X!, Y!), 2
 NEXT R!
 LINE (215, 391)-(275, 391), 2              '←直線のメニュー
 LINE (285, 397)-(305, 385), 2              '←折線のメニュー
 LINE -(325, 397), 2
 LINE -(345, 385), 2
 LINE (355, 386)-(415, 396), 2, B           '←長方形のメニュー
 CIRCLE (455, 391), 23, 2, , , .3           '←楕円のメニュー
 LINE (495, 386)-(555, 396), 2, B           '←色塗のメニュー
 CIRCLE (525, 391), 18, 2, , , .35
 PAINT (496, 387), 4, 2                     '←色塗
 PAINT (525, 391), 5, 2
 PAINT (554, 395), 6, 2
END SUB
```

（３）図形を選択するサブプロシージャ

```
SUB ZukeiSentaku          '←図形を選択するサブプロシージャ
 UEVENT ON                          '←割り込みをＯＮ
 MouseBorder 385, 70, 399, 560      '←マウスの移動範囲設定
 DO
  MousePush 0, Y, X, Hidari, Migi
                   '←左ボタンを押した時点のマウスの位置等の取得
 LOOP UNTIL Hidari                  '←左ボタンを押すと次へ進む
 DO
  MousePoll YY, XX, Hidari, Migi  '←マウスの現在時点の位置等を取得
 LOOP UNTIL NOT Hidari              '←左ボタンを離すと次へ進む
 FOR Z = 1 TO 7
  IF X > Z * 70 AND X < Z * 70 + 70 THEN     '←選択を判定する
```

```
    EXIT FOR                  '←選択されたらNEXT・・・FORの外へ出る
  END IF
NEXT Z
MouseBorder 0, 0, 370, 639         '←マウスの移動範囲を設定
LINE (70, 383)-(560, 399), 0, BF   '←メニューを消去
UEVENT OFF                         '←割り込みをＯＦＦ
SELECT CASE Z                      '←選択により
  CASE 1: Ten                                  '←1ならば点
  CASE 2: TenRenzoku                           '←2ならば点の連続
  CASE 3: Tyokusen                             '←3ならば直線
  CASE 4: Oresen                               '←4ならば折線
  CASE 5: Tyouhoukei                           '←5ならば長方形
  CASE 6: Daen                                 '←6ならば楕円
  CASE 7: IroNuri                              '←7ならば色塗
END SELECT
END SUB
```

（４）色の選択メニューの表示及び色を選択するサブプロシージャ

［色を選択するサブプロシージャ］

```
SUB IroMenu
VIEW PRINT 23 TO 25
CLS 2
FOR I = 0 TO 7                          '←色のメニュー表示
 LINE (320 + (I * 30), 385)-(350 + (I * 30), 395), I, BF
NEXT I
LINE (320, 385)-(530, 395), 7, B        '←メニューの枠の表示
END SUB
```

［色を選択するサブプロシージャ］

```
SUB Iro (C)
LOCATE 23, 5: PRINT "色を選択してください。"
MouseBorder 385, 320, 399, 560   '←マウスの移動範囲をメニュー上に設定
DO                               '←繰り返す
 MousePush 0, Y, X, Hidari, Migi
                                 '←左ボタンを押した時点の位置を取得
LOOP UNTIL Hidari                '←左ボタンが押されるまで繰り返す
FOR C = 0 TO 7                   '←色番号0から7まで繰り返す
 IF X > 320 + C * 30 AND X < 350 + C * 30 THEN   '←範囲内ならば
    EXIT FOR                '←for・・・nextの外へ出る
 END IF
NEXT C
DO
 MousePoll Y, X, Hidari, Migi    '←マウスの現在時点の位置等を取得
```

```
LOOP UNTIL NOT Hidari                '←左ボタンを離してから次に進む
MouseBorder 0, 0, 360, 639          '←マウスの移動範囲を描画画面上に設定
MouseLocate 200, 320                 '←マウスを画面中央へ移動
CLS 2                                '←文字画面を消去
LINE (320, 385)-(560, 395), 0, BF    '←色のメニューを消去
END SUB
```

（5）線の選択メニューの表示及び線を選択するサブプロシージャ

［線の種類のメニュー表示のサブプロシージャ］

```
SUB SensyuMenu
VIEW PRINT 23 TO 25 '←テキスト画面の２３行目から２５行目まで使用する
LINE (100, 390)-(150, 390), 7                '←実線メニューの表示
LINE (150, 390)-(200, 390), 7, , &HF0F0      '←点線メニューの表示
LINE (200, 390)-(250, 390), 7, , &HFF99      '←破線メニューの表示
LINE (250, 385)-(300, 395), 7, BF           '←塗りつぶしメニューの表示
END SUB
```

［線の種類を選択するサブプロシージャ］

```
SUB Sensyu (S)
CLS 2                              '←テキスト画面（文字の画面）の消去
LOCATE 23, 5: PRINT "線種等を選択してください。"
MouseBorder 385, 100, 399, 300     '←マウスの移動範囲をメニュー上に設定
 DO                                '←繰り返す
  MousePush 0, Y, X, Hidari, Migi
                                   '←左ボタンを押した時点の位置を取得
LOOP UNTIL Hidari                  '←左ボタンが押されるまで繰り返す
FOR S = 0 TO 3                     '←線の番号の0から3まで繰り返す
 IF X > 100 + S * 50 AND X < 150 + S * 50 THEN    '←範囲内ならば
    EXIT FOR                    '←FOR・・・NEXTの外へ出る
 END IF
NEXT S
DO
 MousePoll Y, X, Hidari, Migi        '←マウスの現在位置等の取得
LOOP UNTIL NOT Hidari                  '←左ボタンを離してから次に進む
MouseBorder 0, 0, 360, 639         '←マウスの移動範囲を描画画面上に設定
MouseLocate 200, 320                   '←マウスを画面中央へ移動
CLS 2                                  '←文字の消去
LINE (100, 385)-(300, 395), 0, BF      '←メニューの消去
END SUB
```

（６）点を吹き付けて描くサブプロシージャ

```
SUB Ten                    '←点を吹き付けるサブプロシージャ
 IroMenu                   '←色を選択するメニューの画面表示
 Iro C                     '←色の選択
RANDOMIZE TIMER            '←乱数系列の設定
DO
 DO
    MousePush 0, Y, X, Hidari, Migi
                          '←左ボタンを押した時点のマウスの位置等を取得
    IF Migi THEN          '←右ボタンを押すと
        EXIT SUB          '←サブプロシージャを出る
    END IF
 LOOP UNTIL Hidari        '←左ボタンを押すと次へ
 DO
    MousePoll YY, XX, Hidari, Migi    '←マウスの現在位置等の取得
 LOOP UNTIL NOT Hidari                 '←左ボタンを離すと次へ
    FOR I = 1 TO 20                      '←２０個の点を表示
    Q! = RND * 2 * 3.141593           '←角度
    R! = 20 * RND                     '←半径
    XT! = COS(Q!) * R! + X            '←X座標
    YT! = SIN(Q!) * R! + Y            '←Y座標
    PSET (XT!, YT!), C                '←点の表示
 NEXT I
LOOP
END SUB
```

１個の点を描くのではなく、点を吹き付けるようにしました。ここでは、２０個の点を表示するようにしました。

Q! = RND * 2 * 3.141593は、乱数で角度を決めます。角度Q!の単位はラジアンなので、360度は２×πなので、これに乱数を乗じます。乱数は、0<RND<1なので、０度から３６０度までの間のさまざまの角度を得ることができます。

R! = 20 * RNDは、０から２０までの範囲で半径を決めます。これで、半径２０の円内に点を均等に吹き付けることができます。**XT! = COS(Q!) * R! + X** 及び **YT! = SIN(Q!) * R! + Y** は、画面上の座標位置を計算しています。

（７）連続した点を描くサブプロシージャ

```
SUB TenRenzoku              '←連続した点を表示するサブプロシージャ
 IroMenu                    '←色を選択するメニューの表示
 Iro C                      '←色を選択する
DO
 DO
    MousePoll Y, X, Hidari, Migi  '←マウスの左ボタンが押された位置を取得
    IF Migi THEN           '←右ボタンを押すと
        EXIT SUB           '←サブプロシージャを出る
```

```
    END IF
  LOOP UNTIL Hidari         '←左ボタンが押されるまで繰り返す
  PSET (X, Y), C            '←点を描く
LOOP
END SUB
```

（8）直線を描くサブプロシージャ

```
SUB Tyokusen                      '←直線を表示するサブプロシージャ
SensyuMenu                        '←線の種類のメニューを表示する
Sensyu S                          '←線の種類を選択する
IroMenu                           '←色のメニューを表示する
Iro C                             '←色を選択する
DO
    CLS 2                         '←テキスト画面の画面消去
 LOCATE 23, 5: PRINT "直線の始点を指示してください。"
 DO
    MousePush 0, Y1, X1, Hidari, Migi  '←左ボタンを押した時点の位置を返す
    IF Migi THEN                  '←右ボタンを押すと
        EXIT SUB                  '←サブプロシージャを出る
    END IF
 LOOP UNTIL Hidari                '←左ボタンを押すまで繰り返す
 DO
    MousePoll Y, X, Hidari, Migi  '←マウスの位置等を取得する
 LOOP UNTIL NOT Hidari            '←左ボタンを離してから次に進む
 CLS 2                            '←テキスト画面の画面消去
 LOCATE 23, 5: PRINT "直線の終点を指示してください。"
 DO
        MousePoll Y3, X3, H3, M3  '←マウスの位置等を取得する
    SCREEN 0, , 1, 1              '←第二画面を使う
    LINE (X1, Y1)-(X3, Y3), C     '←直線を描く
    FOR I = 1 TO 1000: NEXT       '←時間をおく
    LINE (X1, Y1)-(X3, Y3), 0     '←直線を消す
    SCREEN 0, , 0, 0              '←第一画面を使う
    MousePush 0, Y2, X2, Hidari, Migi  '←左ボタンを押した時点の位置を返す
 LOOP UNTIL Hidari                '←左ボタンを押すまで繰り返す
 SELECT CASE S
    CASE 0: LINE (X1, Y1)-(X2, Y2), C              '←直線を実線で描く
    CASE 1: LINE (X1, Y1)-(X2, Y2), C, , &HF0F0    '←直線を点線で描く
    CASE 2: LINE (X1, Y1)-(X2, Y2), C, , &HFF99    '←直線を破線で描く
    CASE 3: LINE (X1, Y1)-(X2, Y2), C              '←直線を実線で描く
 END SELECT
 DO
    MousePoll Y, X, Hidari, Migi  '←マウスの位置等を取得する
 LOOP UNTIL NOT Hidari            '←左ボタンを離してから次に進む
 LOOP                             '←繰り返す
 END SUB
```

（9）折れ線を描くサブプロシージャ

```
SUB Oresen                    '←連続折れ線を描くサブプロシージャ
SensyuMenu                    '←線の種類を選択するメニュー表示
Sensyu S                      '←線の種類を選択する
IroMenu                       '←色を選択するメニュー表示
Iro C                         '←色を選択する
CLS 2                         '←テキスト画面の画面消去
LOCATE 23, 5: PRINT "折線の始点を指示してください。"
DO
    MousePush 0, Y1, X1, Hidari, Migi   '←左ボタンを押した時点の位置を返す
    IF Migi THEN              '←右ボタンを押すと
        EXIT SUB              '←サブプロシージャを出る
    END IF
LOOP UNTIL Hidari             '←左ボタンを押すまで繰り返す
DO
    MousePoll Y, X, Hidari, Migi   '←マウスの現在時点の位置等を取得
    LOOP UNTIL NOT Hidari          '←左ボタンを離してから次に進む
DO
    CLS 2                          '←テキスト画面の画面消去
    LOCATE 23, 5: PRINT "折線の終点を指示してください。"
    DO
        MousePoll Y3, X3, H3, M3   '←マウスの現在時点の位置等を取得
        SCREEN 0, , 1, 1           '←第二画面を使う
        LINE (X1, Y1)-(X3, Y3), C  '←長方形を描く
        FOR I = 1 TO 1000: NEXT    '←時間をおく
        LINE (X1, Y1)-(X3, Y3), 0  '←長方形を消す
        SCREEN 0, , 0, 0           '←第一画面を使う
        MousePush 0, Y2, X2, Hidari, Migi '←左ボタンを押した時点の位置を返す
        IF Migi THEN               '←右ボタンを押すと
            EXIT SUB               '←サブプロシージャを出る
        END IF
    LOOP UNTIL Hidari              '←左ボタンを押すまで繰り返す
    SELECT CASE S
        CASE 0: LINE (X1, Y1)-(X2, Y2), C              '←実線を描く
        CASE 1: LINE (X1, Y1)-(X2, Y2), C, , &HF0F0    '←点線を描く
        CASE 2: LINE (X1, Y1)-(X2, Y2), C, , &HFF99    '←破線を描く
        CASE 3: LINE (X1, Y1)-(X2, Y2), C              '←実線を描く
 END SELECT
 DO
    MousePoll Y, X, Hidari, Migi   '←マウスの現在時点の位置等を取得
 LOOP UNTIL NOT Hidari             '←左ボタンを離してから次に進む
 X1 = X2: Y1 = Y2                  '←直線の終点を次の直線の始点とする
 LOOP                              '←繰り返す
END SUB
```

（１０）長方形を描くサブプロシージャ

```
SUB Tyouhoukei                    '←長方形のサブプロシージャ
SensyuMenu                        '←線種を選択するメニュー表示
Sensyu S                          '←線種を選択する
IroMenu                           '←色を選択するメニュー表示
Iro C                             '←色を選択する
DO
 CLS 2                        '←テキスト画面を消去
 LOCATE 23, 5: PRINT "長方形の頂点を指示してください。"
 DO
    MousePush 0, Y1, X1, Hidari, Migi     '←左ボタンを押した時点の位置を返す
    IF Migi THEN              '←右ボタンを押すと
        EXIT SUB              '←サブプロシージャを出る
    END IF
 LOOP UNTIL Hidari            '←左ボタンを押すまで繰り返す
 DO
    MousePoll Y, X, Hidari, Migi      '←現在のマウスの位置等を取得
 LOOP UNTIL NOT Hidari                '←左ボタンを離してから次に進む
 CLS 2
 LOCATE 23, 5: PRINT "長方形の向いの頂点を指示してください。"
 DO
    MousePoll Y3, X3, H3, M3          '←現在のマウスの位置等を取得
    SCREEN 0, , 1, 1                        '←第二画面を使う
    LINE (X1, Y1)-(X3, Y3), C, B          '←長方形を描く
    FOR I = 1 TO 1000: NEXT               '←時間をおく
    LINE (X1, Y1)-(X3, Y3), 0, B          '←長方形を描く
    SCREEN 0, , 0, 0                      '←第一画面を使う
    MousePush 0, Y2, X2, Hidari, Migi     '←左ボタンを押した時点の位置を返す
 LOOP UNTIL Hidari                '←左ボタンを押すまで繰り返す
 SELECT CASE S
    CASE 0: LINE (X1, Y1)-(X2, Y2), C, B            '←実線で描く
    CASE 1: LINE (X1, Y1)-(X2, Y2), C, B, &HF0F0    '←点線で描く
    CASE 2: LINE (X1, Y1)-(X2, Y2), C, B, &HFF99    '←破線で描く
    CASE 3: LINE (X1, Y1)-(X2, Y2), C, BF           '←塗りつぶして描く
 END SELECT
 DO
    MousePoll Y, X, Hidari, Migi      '←現在のマウスの位置等を取得
 LOOP UNTIL NOT Hidari                  '←左ボタンを離してから次に進む
LOOP                                  '←繰り返す
END SUB
```

（１１）楕円を描くサブプロシージャ

```
SUB Daen                          '←楕円のサブプロシージャ
IroMenu                                 '←色の選択メニュー
Iro C                                   '←色の選択
```

```
DO
    CLS 2                              '←テキスト画面の消去
    LOCATE 23, 5: PRINT "楕円の中心を指示してください。"
    DO
        MousePush 0, Y1, X1, Hidari, Migi  '←左ボタンを押した時点の位置を返す
        IF Migi THEN                   '←右ボタンを押すと
            EXIT SUB                   '←サブプロシージャを出る
        END IF
    LOOP UNTIL Hidari                  '←左ボタンを押すまで繰り返す
    DO
        MousePoll Y, X, Hidari, Migi     '←マウスの現在の位置等を取得
    LOOP UNTIL NOT Hidari                '←左ボタンを離してから次に進む
    CLS 2                                '←テキスト画面の消去
 LOCATE 23, 5: PRINT "楕円の半径を指示してください。"
 DO
    MousePoll Y3, X3, H3, M3          '←現在のマウスの位置等を取得
    SCREEN 0, , 1, 1                  '←第二画面の使用
    IF X3 <> X1 THEN                  '↓仮の楕円を描く
        CIRCLE (X1, Y1), SQR((X1 - X3) ^ 2 + (Y1 - Y3) ^ 2),
         <前行続き> C, , , ABS((Y1 - Y3) / (X1 - X3))
        FOR I = 1 TO 1000: NEXT    '←時間をおいて↓仮の楕円を消す
        CIRCLE (X1, Y1), SQR((X1 - X3) ^ 2 + (Y1 - Y3) ^ 2),
         <前行続き> 0, , , ABS((Y1 - Y3) / (X1 - X3))
    END IF
    SCREEN 0, , 0, 0                  '←第一画面の使用
    MousePush 0, Y2, X2, Hidari, Migi   '←左ボタンを押した時点の位置を返す
 LOOP UNTIL Hidari                    '←左ボタンを押すまで繰り返す
 R = SQR((X1 - X2) ^ 2 + (Y1 - Y2) ^ 2)     '←半径の計算
 IF X1 <> X2 THEN
    Hi! = ABS((Y1 - Y2) / (X1 - X2))        '←長径と短径の比の計算
 ELSE
    Hi! = 1000               '←分母(X1-X2)が0の場合は比は1000とする
 END IF
 DO
    MousePoll Y, X, Hidari, Migi
                              '←マウスの現在時点の位置等を取得
 LOOP UNTIL NOT Hidari        '←左ボタンを離してから次に進む
 CLS 2                        '←テキスト画面の消去
 LOCATE 23, 5: PRINT "         円弧の始点を指示してください。"
 DO
    MousePoll Y3, X3, H3, M3          '←マウスの現在の位置等を取得
    SCREEN 0, , 1, 1                  '←第二画面の使用
    IF X3 <> X1 THEN
        CIRCLE (X1, Y1), R, C, Enko!(X1, Y1, X3, Y3), , Hi!
                                         '←仮の円弧を描く
        FOR I = 1 TO 1000: NEXT   '←表示する時間をおく
        CIRCLE (X1, Y1), R, 0, Enko!(X1, Y1, X3, Y3), , Hi!
                                         '←仮の円弧を消す
        END IF
```

```
        SCREEN 0, , 0, 0                        '←第一画面の使用
      MousePush 0, Y2, X2, Hidari, Migi
                                              '←左ボタンを押した時点の位置を返す
   LOOP UNTIL Hidari                          '←左ボタンを押すまで繰り返す
   ST! = Enko!(X1, Y1, X2, Y2)                '←円弧の始点の計算
   DO
      MousePoll Y, X, Hidari, Migi            '←マウスの現在の位置等を取得
   LOOP UNTIL NOT Hidari                      '←左ボタンを離してから次に進む
   CLS 2
   LOCATE 23, 5: PRINT "       円弧の終点を指示してください。"
   DO
      MousePoll Y3, X3, H3, M3                '←マウスの現在の位置等を取得
      SCREEN 0, , 1, 1                        '←第二画面を使う
      IF X3 <> X1 THEN
         CIRCLE (X1, Y1), R, C, ST!, Enko!(X1, Y1, X3, Y3), Hi!
                                              '←仮の円弧を描く
         FOR I = 1 TO 1000: NEXT              '←時間をおく
         CIRCLE (X1, Y1), R, 0, ST!, Enko!(X1, Y1, X3, Y3), Hi!
                                              '←仮の円弧を消す
      END IF
      SCREEN 0, , 0, 0
      MousePush 0, Y2, X2, Hidari, Migi       '←左ボタンを押した時点の位置を返す
   LOOP UNTIL Hidari                          '←左ボタンを押すまで繰り返す
   EN! = Enko!(X1, Y1, X2, Y2)                '←円弧の終点の計算
   DO
      MousePoll Y, X, Hidari, Migi            '←マウスの現在の位置等を取得
   LOOP UNTIL NOT Hidari                      '←左ボタンを離してから次に進む
   IF ST! = EN! THEN
      ST! = 0: EN! = 3.141593 * 1.9999999#
                                              '←すべての円弧を描く場合の角度
   END IF
   CIRCLE (X1, Y1), R, C, ST!, EN!, Hi!       '←円・楕円・円弧を描く
   DO
      MousePoll Y, X, Hidari, Migi            '←マウスの現在の位置等を取得
   LOOP UNTIL NOT Hidari                      '←左ボタンを離してから次に進む
  LOOP                                        '←繰り返す
END SUB
```

（１２）円弧の始点と終点を計算する関数のサブプロシージャ

```
FUNCTION Enko! (X1, Y1, XX, YY)
                    '←円弧の始点と終点を計算する関数のサブプロシージャ
IF XX <> X1 AND X1 < XX AND Y1 > YY THEN           '←第１象限の場合
 Enko! = ATN((Y1 - YY) / (XX - X1)): EXIT FUNCTION
END IF
IF YY <> Y1 AND X1 > XX AND Y1 > YY THEN           '←第２象限の場合
 Enko! = ATN((X1 - XX) / (Y1 - YY)) + (3.141593 / 2): EXIT FUNCTION
```

```
END IF
IF XX <> X1 AND X1 > XX AND Y1 < YY THEN          '←第３象限の場合
 Enko! = ATN((YY - Y1) / (X1 - XX)) + 3.141593: EXIT FUNCTION
END IF
IF YY <> Y1 AND X1 < XX AND Y1 < YY THEN          '←第４象限の場合
 Enko! = ATN((XX - X1) / (YY - Y1)) + 3.141593 * 1.5: EXIT FUNCTION
END IF
IF XX = X1 AND Y1 > YY THEN                 '←90度の場合
 Enko! = 3.141593 / 2: EXIT FUNCTION
END IF
IF XX = X1 AND Y1 < YY THEN                 '←270度の場合
 Enko! = 3.141593 * 1.5: EXIT FUNCTION
END IF
IF YY = Y1 AND X1 > XX THEN                 '←180度の場合
 Enko! = 3.141593: EXIT FUNCTION
END IF
IF YY = Y1 AND X1 < XX THEN                 '←0度の場合
 Enko! = 0: EXIT FUNCTION
END IF
END FUNCTION
```

（１３）色を塗るサブプロシージャ

```
SUB IroNuri                  '←色を塗るサブプロシージャ
 IroMenu                                '←色を選択するメニュー表示
LOCATE 23, 40: PRINT "（塗る色の指定）";
 Iro C1                                 '←塗る色の選択
 IroMenu                                '←色を選択するメニュー表示
LOCATE 23, 40: PRINT "（境界色の指定）";
 Iro C2                                  '←境界色の選択
DO
 DO
    MousePush 0, Y, X, Hidari, Migi
                             '←左ボタンを押した時点のマウスの位置等を取得
    IF Migi THEN                       '←右を押せば
        EXIT SUB                       '←サブプロシージャを出る
    END IF
 LOOP UNTIL Hidari                      '←左ボタンを押せば次へ
 DO
    MousePoll YY, XX, Hidari, Migi '←マウスの現在時点の位置等を取得
 LOOP UNTIL NOT Hidari                  '←左ボタンを離すと次へ
 PAINT (X, Y), C1, C2                   '←色を塗る
LOOP
END SUB
```

第⑨章

通信ソフト

パソコン通信の基礎知識と通信ソフトのプログラミングを学びます。

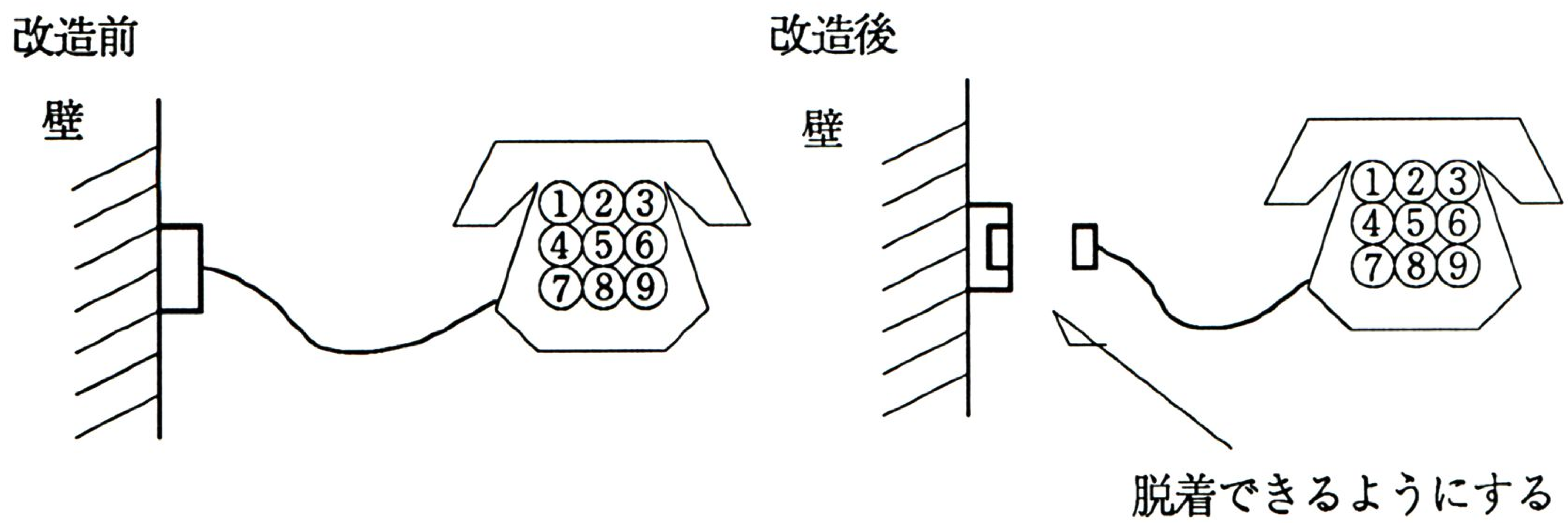
9．1　パソコン通信の準備

パソコン通信を始めるには、パソコンの外にモデムという機器を用意する必要があります。モデムは、電話機とパソコンの間の、仲介役の機器です。なにしろ、パソコンはパソコン語を電話は電話語を、それぞれの都合で話をするので、モデムは、双方の言葉の翻訳や交通整理をする重要な仕事をします。

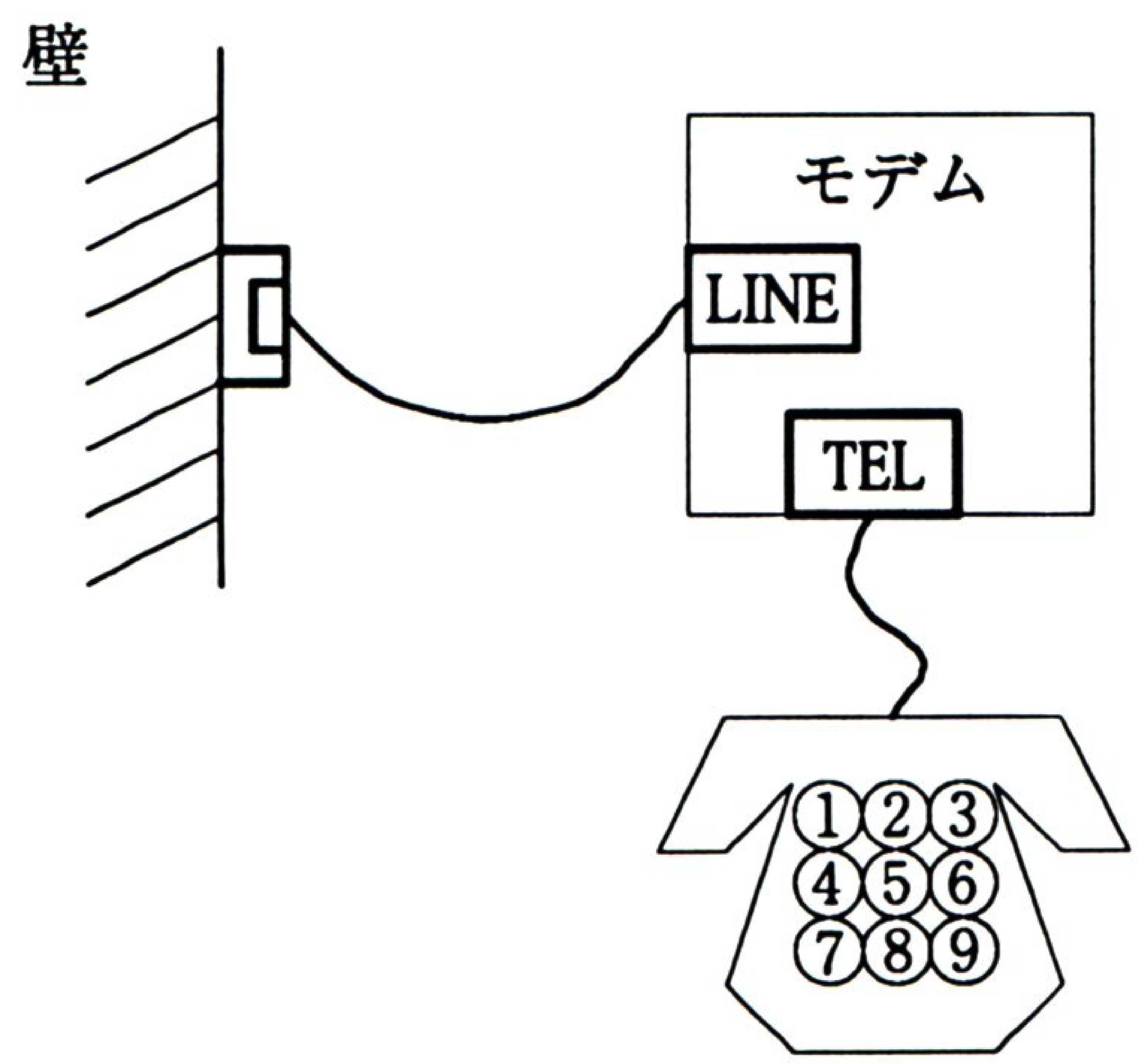
かつては、モデムはたいへん高価な機器でしたが、今では、高性能でなおかつ価格も買いやすいようになったので、思い切って買われたらいかがでしょうか。

モデムとパソコンに結線して使用するまでの手順は次のとおりです。

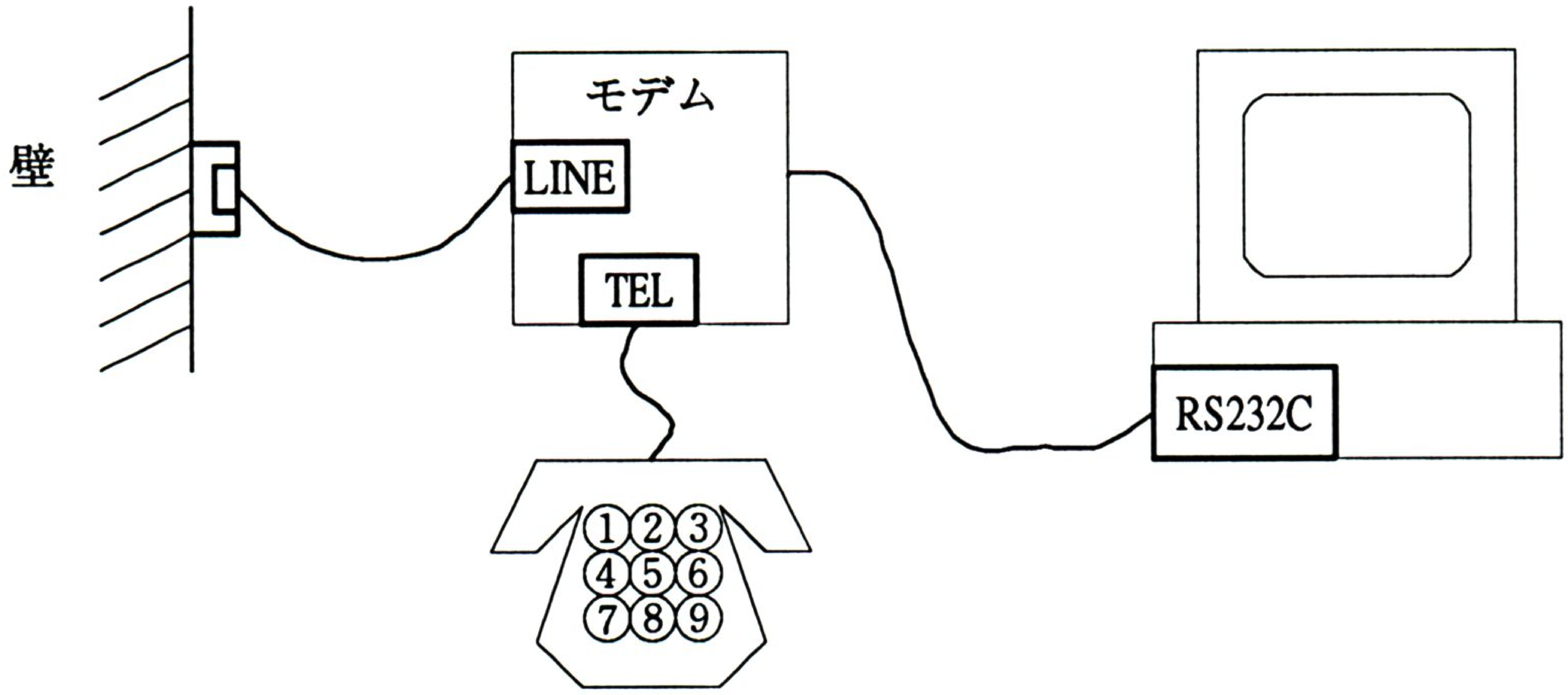
（1）電話をモジュラージャックで接続できるようにする。（電話工事業者に依頼する。）



（2）電話回線とモデムのLINEを接続し、電話機とモデムのTELを接続する。



（３）モデムとパソコンのＲＳ２３２Ｃインタフェースとを接続する。



（４）パソコン及びモデムの電源コードを接続して、電源をＯＮにする。

なお、モデムの使用に際して、モデムのディップスイッチをセットする必要がありますが、出荷時のままで使える場合が多いようです。モデムのマニュアルで確認してください。

モデムを使うためには、モデムに命令を送る必要があります。この命令が、モデムのメーカーごとに違っていては不便なので、一般にＡＴコマンドという方式を使っております。ＡＴコマンド方式は米国でもっとも普及したコマンド体系なので、日本のメーカーもこれを採用したようです。

パソコンでダイヤルして、パソコン通信のホスト局の電話に接続する手順は次のとおりです。

（１）ＲＳ２３２Ｃの通信ポートをランダムアクセスファイルとしてオープンする。

```
OPEN "COM1:1200,N,8,1,CD0,DS0" FOR RANDOM AS #1 LEN = 256
```

- 256 → レコードの長さが２５６バイト
- AS #1 → ファイル番号が#1
- FOR RANDOM → ランダムアクセスファイル
- CD0,DS0 → 回線上のタイムアウトのチェックをしない
- 1 → ストップビットの長さ（1 か 2 --> データの区切りのためのビット数）
- 8 → データのビットの長さ
- N → パリティチェックの種類　（なし-->N 偶数パリティ-->E 奇数パリティ-->0）
- 1200 → 通信速度（1秒間に送るビット数、300,1200,2400）
- COM1 → 通信ポートのファイル名（COM1、COM2）

（２）モデムを初期化する。
ATZをモデムに送る。　　　　　PRINT #1,"ATZ"
すると、モデムからOKという返事が返ってくる。
（３）ダイヤルする。
例えば　　　パソコン通信のホスト局の電話番号　　012-345-6789
　　　　　　自分の電話がパルスダイヤル式の電話
ATDP012-345-6789をモデムに送る。
PRINT #1,"ATDP012-345-6789"　（ﾄｰﾝﾀﾞｲﾔﾙ　"ATDT012-345-6789"）
するとダイヤルを開始し、接続すると、CONNECTと返事が返ってくる。

基本的な通信プログラムは、次のとおりです。このプログラムは、まだ不完全ですが、小さいプログラムなので、概略を知るには格好です。

ローカルのパソコン通信のホスト局でＭＮＰ（後で解説します。）を使っていない場合は、この程度でも接続できます。しかし、ＰＣ－ＶＡＮやＮＩＦＴＹ　Ｓｅｒｖｅの場合は、しっかりプログラムをつくらないと接続できないようです。

```
DEFINT A-Z
'通信1        ****通信ソフトの基本****        (TUUSIN1.BAS)
CLS                                      '←画面消去
LOCATE 1, 1, 1, 14, 15                   '←カーソルを表示する
OPEN "COM1:1200,N,8,1,CD0,DS0" FOR RANDOM AS #1 LEN = 256
                                         '←通信　回線のファイルを開く
PRINT #1, "ATZ"                          '←モデムの初期化
MM$ = ""
DO
    IF LOC(1) THEN       '←通信のファイルに入力待ちの文字があるならば
        M$ = INPUT¥(LOC(1), #1)          '←文字数分を読む
        PRINT M$;                        '←画面へ表示する
        MM$ = MM$ + M$                   '←文字を累積する
    END IF
LOOP UNTIL INSTR(MM$, "OK")              '←OKならば次へ
FOR I = 1 TO 10000: NEXT                 '←時間をおく
PRINT #1, "ATDP012-345-6789"             '←電話番号をダイヤルする
MM$ = ""
DO
    IF LOC(1) THEN       '←通信のファイルに入力待ちの文字があるならば
        M$ = INPUT¥(LOC(1), #1)          '←文字数分を読む
        PRINT M$;                        '←画面へ表示する
        MM$ = MM$ + M$                   '←文字を累積する
    END IF
LOOP UNTIL INSTR(MM$, "CONNECT")         '←CONNECTならば次へ
DO
    K$ = INKEY$                          '←1文字入力
    IF K$ <> "" THEN                     '←文字が入力されていれば
        PRINT #1, K$;                    '←送信する
```

```
    END IF
    IF LOC(1) THEN                         '←入力待ちの文字があるならば
        M$ = INPUT¥(LOC(1), #1)            '←文字数分を読む
        PRINT M$;                          '←画面へ表示
    END IF
LOOP
END
```

LOC(1) によって、ファイル番号1のバッファーに存在する文字の数を取得できるので、IF LOC(1) THEN は、通信ポートに文字が受信してあれば・・ということになります。

M$ = INPUT¥(LOC(1), #1)によって、ファイル番号1バッファーに存在する文字の数の分だけ、M$に移動します。=は、ここでけは、複写ではなく移動なので、読み取れば消えるようになっています。

MM$ = MM$ + M$ で、通信ポートから得た文字を蓄積しておいて、OKやCONNECTの返事がくるかどうかを判定します。

９．２　ＡＴコマンドと送受信

前節では、ＡＴコマンドの使用を最小限にしましたが、確実にモデムを作動させるためには多数のコマンドを送る必要があります。そこで必要とするコマンドを次にまとめました。

（１）モデムを初期化する。
　PRINT #1,"ATZ"

（２）電話の方式を設定する。
　トーンダイヤル方式の場合　PRINT #1,"ATT"
　パルスダイヤル方式の場合
　１秒間に１０パルスの場合　PRINT #1,"ATP1"　（普通はこれを使います。）
　１秒間に２０パルスの場合　PRINT #1,"ATP2"

（３）通信規格を設定する。
　ＣＣＩＴＴ規格の場合　PRINT #1,"ATB0"　（普通はこれを使います。）
　ＢＥＬＬ規格の場合　PRINT #1,"ATB1"

（４）接続した時に通信速度の表示をする。
　PRINT #1,"ATX4"　（普通はこれを使います。）

（５）ＭＮＰ接続及びＭＮＰクラスを表示する。
　表示しない場合　PRINT #1,"AT¥V0"
　表示する場合　PRINT #1,"AT¥V2"

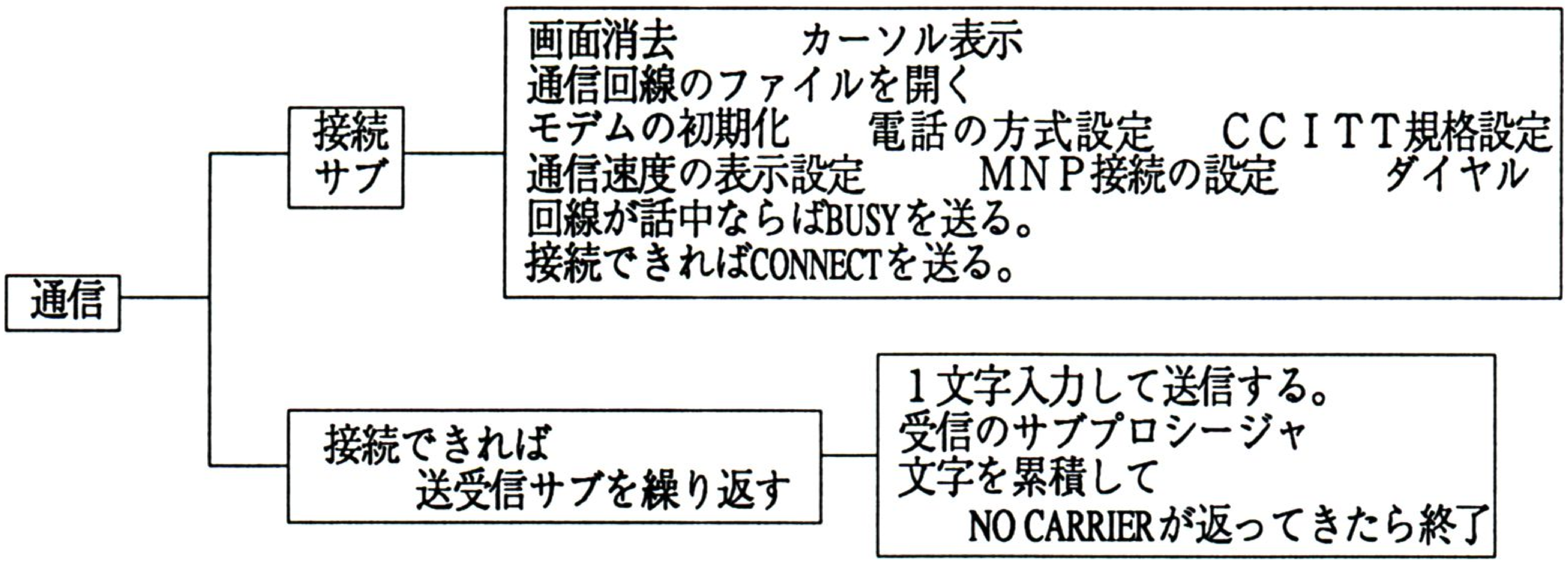
（６）ＭＮＰの接続を設定する。
　接続しない場合　PRINT #1,"AT¥N0"
　接続する場合　PRINT #1,"AT¥N3"

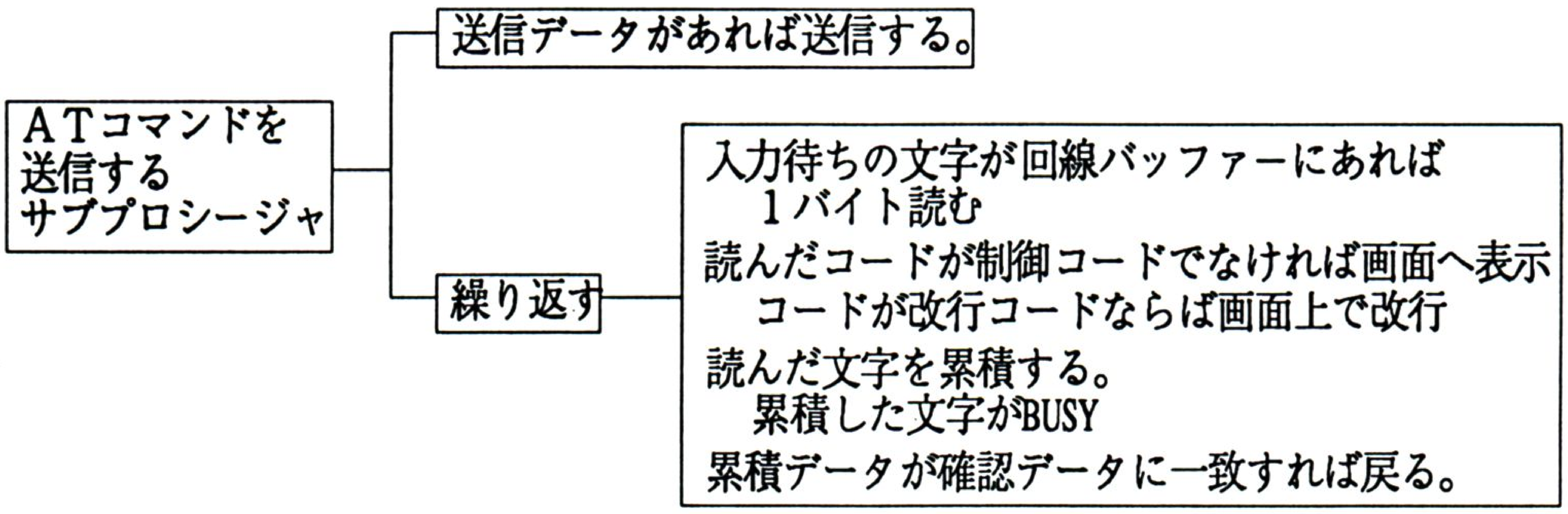
（７）ダイヤルする。
　PRINT #1,"ATD0123-45-6789"　（普通はこれを使います。）

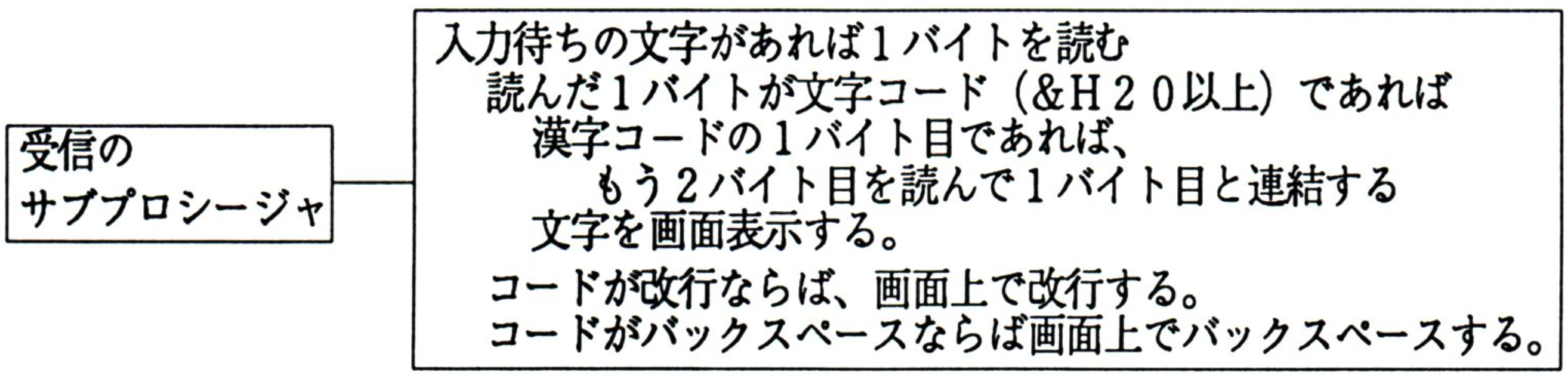
（８）回線を切る。
　PRINT #1,"ATH"

（９）オンラインモードからコマンドモードへ切り替える。
　１分間なにもしない時間が経過した後、PRINT #1,"AT+++";

プログラムの構造の概要は次ページのとおりです。







（１）メインプロシージャ

```
DEFINT A-Z
DECLARE SUB BackSpace ()
DECLARE SUB Setuzoku (KakuninData$)
DECLARE SUB SouZyusin ()
DECLARE SUB Zyusin (M$)
DECLARE SUB AtSousin (SousinData$, KakuninData$)
'通信    ****  ＡＴコマンドと送受信  ****  (TUUSIN2.BAS)
Setuzoku KakuninData$                          '←接続する
IF KakuninData$ = "CONNECT" THEN               '←CONNECTと返ってきたら
    SouZyusin                                   '←送受信を開始
END IF
END
```

（2）接続のサブプロシージャ

```
SUB Setuzoku (KakuninData$)          '←接続のサブプロシージャ
CLS                                  '←画面消去
LOCATE 1, 1, 1, 14, 15               '←カーソルを表示する
FILEMEI$ = "COM1:1200,N,8,1,CD0,DS0"
OPEN FILEMEI$ FOR RANDOM AS #1 LEN = 256  '←通信回線のファイルを開く
Kaigyou$ = CHR$(&HD)                 '←改行
LineFeed$ = CHR$(&HA)                '←ラインフィード
OK$ = "OK" + Kaigyou$ + LineFeed$    '←OK$の設定
AtSousin "ATZ", OK$                  '←モデムの初期化
AtSousin "ATP1", OK$                 '←自分の電話の方式
AtSousin "ATB0", OK$                 '←ＣＣＩＴＴ規格
AtSousin "ATX4", OK$                 '←接続後の通信速度の表示
Mnp$ = "0"                 '←1にするとＭＮＰ接続　0にすると非接続
SELECT CASE Mnp$
    CASE "0": AtSousin "AT¥V0", OK$  '←ＭＮＰ接続しない
              AtSousin "AT¥N0", OK$
    CASE "1": AtSousin "AT¥V2", OK$  '←ＭＮＰ接続する
              AtSousin "AT¥N3", OK$
END SELECT
KakuninData$ = "CONNECT"
AtSousin "ATD0123-45-6789", KakuninData$ '←センターの電話をダイヤルする
IF KakuninData$ = "BUSY" THEN        '←話中のときは
    CLOSE                            '←ファイルを閉じて
    PRINT " 話中です。 何かキーを押してください。"
    DO: LOOP UNTIL INKEY$ <> ""      '←何かキーを押したら
    EXIT SUB                         '←はじめに戻る
END IF
AtSousin "", Kaigyou$ + LineFeed$    '←改行とラインフードの確認
KakuninData$ = "CONNECT"   '←確認データを返すためCONNECTをセット
END SUB
```

ＡＴコマンドをモデムに送ると、OK、CONNECT、BUSYなどのメッセージが返ってきます。

さらに、これらに続いて、0D 0A が送られてきます。

0Dは、カーソルを左端に移動する制御コードで、復帰といい、0Aは、カーソルを次の行に移動する制御コードでラインフィードといいます。

実は、PRINT CHR$(&HD);とすると、N88BASICでは、同じ行で復帰だけ行うのですが、Quick BASICでは、次の行に改行するので、0Dをここでは改行と表現しています。そこで、Quick BASICで、送られてくる0D0Aを、PRINT CHR$(&HD); : PRINT CHR$(&HA); とすると２度の改行をすることになります。

ATD（電話番号）でダイヤルして、話中の場合はBUSYが、接続できればCONNECTが返ってきます。これをメインプロシージャへ返すと、メインはこれを判断してCONNECTの場合だけ送受信を開始します。

（3）ATコマンドを送信するサブプロシージャ

```
SUB AtSousin (SousinData$, KakuninData$)     '←送信のサブプロシージャ
IF SousinData$ <> "" THEN                    '←送信データがあれば
    PRINT #1, SousinData$                    '←送信する
END IF
MM$ = ""
DO
    IF LOC(1) THEN                           '←入力待ちの文字があるならば
        M$ = INPUT$(1, #1)                   '←1バイトを読む
        SELECT CASE M$
            CASE IS >= CHR$(&H20)            '←コードが16進数20以上の場合は
                PRINT M$;                    '←画面表示する
            CASE CHR$(13)                    '←コードが改行ならば
                PRINT                        '←画面上で改行する
        END SELECT
        MM$ = MM$ + M$                       '←文字を累積する
    END IF
    IF INSTR(MM$, "BUSY") THEN               '←話中ならば
        KakuninData$ = "BUSY"                '←確認データにBUSYを複写して
        EXIT SUB                             '←サブプロシージャを出る
    END IF
LOOP UNTIL INSTR(MM$, KakuninData$)          '←累積した文字の中に確認する文字があれば
FOR I = 1 TO 10000: NEXT I                   '←時間をおく
END SUB                                      '←サブプロシージャを出る
```

PRINT #1, SousinData$ でATコマンドをモデムへ送ります。モデムから返ってくる文字は、IF LOC(1) THEN で判断して回線のバッファーから、M$ = INPUT$(1, #1) で1バイト分読み込みます。文字のコードは、&H20（16進数で20）以上なので、M$ が16進数02以上ならば、文字を画面へ表示し、0Dならば改行するようにします。

受信した文字は、MM$へ累積して、"OK"や"CONNECT"など確認する文字が返ってくるまで繰り返し、確認できたらこのサブプロシージャを出ます。

また、話中のため、"BUSY"が返ってきたら、これを返します。

（4）送受信するサブプロシージャ

```
SUB SouZyusin                                '←送受信のサブプロシージャ
DO
    K$ = INKEY$                              '←1文字入力
    IF K$ <> "" THEN                         '←もしも入力されれば
        PRINT #1, K$;                        '←送信する
    END IF
    Zyusin M$                                '←受信のサブプロシージャ
    IF M$ = CHR$(&HD) THEN                   '←M$が改行コードならば
        MM$ = ""                             '←MM$をクリア
```

```
    ELSE                                    '←そうでなければ
        MM$ = MM$ + M$                      '←MM$へM$を累積
    END IF
LOOP UNTIL INSTR(MM$, "NO CARRIER")        '←MM$にNO CARRIERがあれば、
CLOSE                                       '←ファイルを閉じて出る
END SUB
```

送信データは、K$ = INKEY$ で１文字入力し、PRINT #1, K$; で送信します。この時、入力して送信した文字を画面には表示しないのですが、送信データは直ちにセンターから返されてくるので、これを受信して表示します。受信の仕事は、受信のサブプロシージャが担当します。

受信したデータは、受信のサブプロシージャからM$として返されてくるので、これを累積して、NO CARRIER（回線切断）を判断して終了とします。

（５）受信のサブプロシージャ

```
SUB Zyusin (M$)                          '←受信のサブプロシージャ
    M$ = ""
    IF LOC(1) THEN                           '←入力待ちの文字があるならば
        M$ = INPUT$(1, #1)                   '←１バイトを読む
        SELECT CASE M$
            CASE IS >= CHR$(&H20)    '←コードが１６進数20以上の場合は
                IF (M$ >= CHR$(&H80) AND M$ <= CHR$(&H9F)) OR
                    (M$ >= CHR   $(&HE0) AND M$ <= CHR$(&HEF)) THEN
                    Kanji$ = M$
                    '↑漢字コードの１バイト目ならば←１バイト目を保存し
                    M$ = INPUT$(1, #1)      '←２バイト目を読み取って
                    M$ = Kanji$ + M$        '←１バイト目と２バイト目を連結する
                END IF
                PRINT M$;                   '←データを画面表示する
            CASE CHR$(13)                   '←コードが改行ならば
                PRINT                       '←画面上で改行する
            CASE CHR$(8)                    '←バックスペースならば
                BackSpace                   '←バックスペースの手続き
        END SELECT
    END IF
END SUB
```

IF LOC(1) THEN で入力待ちの文字が回線のバッファーにあるか判定して、あれば M$ = INPUT$(1, #1) で１バイト分を読み取ります。

アスキーコードは、１６進数で00から1Fまでは、制御コードに使っているので、まず、読み取ったM$が、１６進数の20以上かどうかを判定します。

次に、１バイトが漢字コードの１バイト目かどうかを判定します。これは、パソコン通信では、漢字コードとしてシフトＪＩＳコードを使用している場合が多く、シフ

トＪＩＳコードの場合は、１バイト目のコードが決っているから判断できます。すなわち、シフトＪＩＳの漢字コード表によると、１バイト目は、１６進数で80から9Fまで、及び、E0からEFまでとなっております。

これに対して、ＪＩＳコードを利用する場合は、このような判定はできないので、「次のコードが漢字です。」「漢字コードは終り。」ということを表すコード（ＫＩコード、ＫＯコード）を使う必要があって、少々面倒になります。

漢字コードの１バイト目とわかると、２バイト目を読んで、これを加えて、２バイトの漢字コードにして、画面に表示します。もちろん、漢字コードでなければ、１バイトのままで画面表示します。

制御コードの中でも、復帰コード（&HD）の場合は改行し、バックスペース（&H8）の場合は、バックスペースのサブプロシージャ使って、画面上でバックスペースになるよう表示します。

（６）バックスペースのサブプロシージャ

```
SUB BackSpace
y = CSRLIN: X = POS(0)        '←カーソルの現在位置を取得する
LOCATE y, X - 1               '←左へ移動
PRINT " ";                    '←スペースで１文字を消去
LOCATE y, X - 1               '←左へ移動
END SUB
```

ＭＮＰとは？

パソコン通信で使われている無手順の通信方式では、しばしば伝送エラーを生じます。米国のMicrocom社は、これを避けるために、伝送エラーを起こさないような画期的な通信手順を開発しました。この通信手順をＭＮＰ（Microcom Networking Protocol）といいます。これは、双方のモデムがハードウェア上で、伝送データにエラーチェックのための情報などを加えてフレームという包にして送受信する方法です。また、クラス５以上ではデータの圧縮をして伝送速度を高めています。

これまで、無手順の通信方式を、別名、垂れ流し方式といって、文字化けをしばしば起こす方式とされてきましたが、ＭＮＰの機能をもつモデムを採用することによって、エラーのないデータをより高速に送ることができるようになりました。

9．3　オートログイン

前節のプログラムでは、電話の接続後にＩＤ番号やパスワードなどを入力する必要があります。この部分もプログラムに加えると、プログラムが自動的にセンターのコンピュータとの接続までできるようになります。このことを一般にオートログインといっています。

まず、電話の接続後に送信するデータの内容の表現方法を決めます。

（１）送信するデータをそのまま表記する。

（２）改行コード（&H0D &H0A）は!と表記する。
　　そこで、「ABCリターン」ならば、ABC!と表記する。

（３）プロンプトが送られてから送信する場合は、プロンプトを[]で囲んで表記する。
　　例えば、Enter Connection-ID --->の後にSVCリターンを送信する場合は、[->]SVC!とします。

（４）タイミングを取るために時間をおく必要のある場合は、*を表記します。
　　*がひとつで、１秒以内の時間を待ちます。

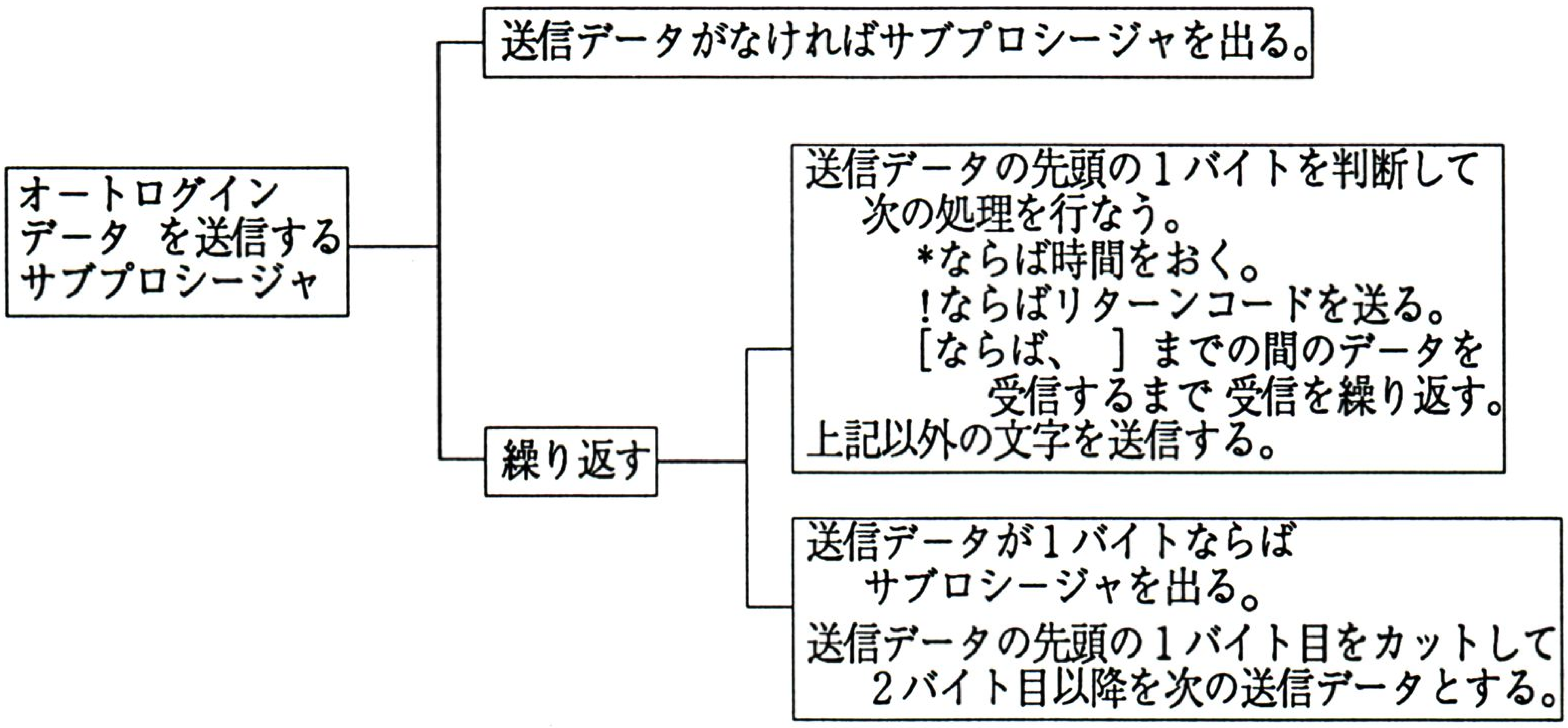
電話の接続後にどのような送信をするかを確かめます。

（１）電話の接続直後に送る信号
　①　一般のパソコン通信の場合は、改行コード（&H0D &H0A）を送ることが多いようです。
　　これを!と表記します。
　②　ＰＣ－ＶＡＮの場合は、VANPCNEC02及び改行コード（&H0D &H0A）を送ります。
　　これをVANPCNEC02!と表記します。
　③　ＮＩＦＴＹ　Ｓｅｒｖｅの場合は、.01+を送ると、改行が返ってきて、さらにEnter Connection-ID --->が送信されるので、SVC及び改行コード（&H0D &H0A）を送ります。
　　これを.01+[->]SVC!と表記します。

（２）ＩＤ番号
　まず、ＩＤ番号のプロンプトが送られてきます。一般には、-->が多いようです。
　これを、[->]と表記します。また、ＰＣ－ＶＡＮの場合は、[:]です。
　ＩＤ番号は、例えば、ABC123!と表記します。

（３）パスワード
　ＩＤ番号の場合と同じです。
　パスワードのプロンプトは、[->]、[:]のように表記します。
　パスワードは、例えば、XYZ987A!と表記します。

オートログインのデータを送るためのサブプロシージャ（Sousin）をつくります。
Sousinのプログラムの構造は、次のとおりです。



（1）メインプロシージャ

```
DEFINT A-Z
DECLARE SUB Setuzoku (KakuninData$)
DECLARE SUB SouZyusin ()
DECLARE SUB Zyusin (M$)
DECLARE SUB BackSpace ()
DECLARE SUB Sousin (SousinData$)
DECLARE SUB AtSousin (SousinData$, KakuninData$)
'通信3    ****  オートログイン  ****     (TUUSIN3.BAS)
Setuzoku KakuninData$                    '←接続する
IF KakuninData$ = "CONNECT" THEN         '←CONNECTとかえってきたら
    SouZyusin                            '←送受信を開始
END IF
END
```

（2）接続のサブプロシージャ

```
SUB Setuzoku (KakuninData$)              '←接続のサブプロシージャ
 CLS 2                                   '←文字画面消去
 LOCATE 2, 1, 1, 14, 15                  '←カーソルを表示する
 FILEMEI$ = "COM1:2400,N,8,1,CD0,DS0"
 OPEN FILEMEI$ FOR RANDOM AS #1 LEN = 256 '←通信回線のファイルを開く
 Kaigyou$ = CHR$(&HD)                    '←改行
 LineFeed$ = CHR$(&HA)                   '←ラインフィード
 OK$ = "OK" + Kaigyou$ + LineFeed$       '←OK$の設定
 AtSousin "ATZ", OK$                     '←モデムの初期化
```

```
AtSousin "ATP1", OK$                          '←自分の電話の方式
AtSousin "ATB0", OK$                          '←ＣＣＩＴＴ規格
AtSousin "ATX4", OK$                          '←接続後の通信速度の表示
Mnp$ = "1"                                    '←ＭＮＰ接続の場合
SELECT CASE Mnp$
    CASE "0": AtSousin "AT¥V0", OK$           '←ＭＮＰ接続しない
              AtSousin "AT¥N0", OK$
    CASE "1": AtSousin "AT¥V2", OK$           '←ＭＮＰ接続する
              AtSousin "AT¥N3", OK$
END SELECT
KakuninData$ = "CONNECT"
AtSousin "ATD0196-51-1101", KakuninData$              '←ダイヤル
IF KakuninData$ = "BUSY" THEN                         '←話中のときは
    CLOSE                                             '←ファイルを閉じて
    PRINT " 話中です。 何かキーを押してください。"
    DO: LOOP UNTIL INKEY$ <> ""                       '←何かキーを押したら
    EXIT SUB                                          '←はじめに戻る
END IF
AtSousin "", Kaigyou$ + LineFeed$             '←改行とラインフィードの確認
Sousin "*.01+[->]SVC!"                        '←接続直後の信号
Sousin "[->]"                                 '←ＩＤ番号のプロンプト
Sousin "ABC012345!"                           '←ＩＤ番号
Sousin "[->]"                                 '←パスワードのプロンプト
Sousin "XY987654!"                            '←パスワード
KakuninData$ = "CONNECT"                      '←確認のためにCONNECTのセット
END SUB
```

（３）ＡＴコマンドを送信するサブプロシージャ

```
SUB AtSousin (SousinData$, KakuninData$)   '←送信のサブプロシージャ
（省略　前節に同じ）
LOOP UNTIL INSTR(MM$, KakuninData$)
                              '←累積した文字の中に確認する文字があれば
FOR I = 1 TO 10000: NEXT I                 '←時間をおく
END SUB                                    '←サブプロシージャを出る
```

（４）オートログインコマンドを送信するサブプロシージャ

```
SUB Sousin (SousinData$)
            '←自動接続のためのコマンドを送信するサブプロシージャ
S$ = SousinData$
IF S$ = "" THEN                            '←送信データがなければ
    EXIT SUB                               '←サブプロシージャを出る
END IF
DO
    SELECT CASE MID$(S$, 1, 1)             '←送信データの１バイト目が
```

```
        CASE "*": T& = TIMER              '←*ならば時間をおく（0.5くらい
            DO: LOOP UNTIL TIMER > T&
        CASE "!": PRINT #1, ""            '←!ならば改行コードを送る
        CASE "["                          '←[ならば
            PR$ = MID$(S$, 2, INSTR(S$, "]") - 2) '←入力待の文字（プロンプト）
            MM$ = ""
            DO
                Zyusin M$                 '←受信の手続きへ
                MM$ = MM$ + M$            '←文字を累積
                M$ = ""
            LOOP UNTIL INSTR(MM$, PR$)
                                  '←累積した文字にプロンプトを含めば次へ
          S$ = MID$(S$, INSTR(S$, "]"), LEN(S$) - INSTR(S$, "]") + 1)
                                  '←S$からプロンプトをカット
        CASE ELSE
        PRINT #1, MID$(S$, 1, 1);            '←送信する
        Zyusin M$                                  '←受信する手続き
    END SELECT
    FOR I = 1 TO 1000: NEXT I              '←時間をおく
    IF LEN(S$) = 1 THEN           '←1バイトならサブプロシージャを出る
        EXIT DO
    END IF
    S$ = MID$(S$, 2, LEN(S$) - 1)    '←送信データの1バイト目をカット
LOOP
END SUB
```

（5）送受信するサブプロシージャ

```
SUB SouZyusin                         '←送受信のサブプロシージャ
    (省略　前節に同じ)
CLOSE                                 '←ファイルを閉じて出る
END SUB
```

（6）受信のサブプロシージャ

```
SUB Zyusin (M$)                   '←受信のサブプロシージャ
    (省略　前節に同じ)
END SUB
```

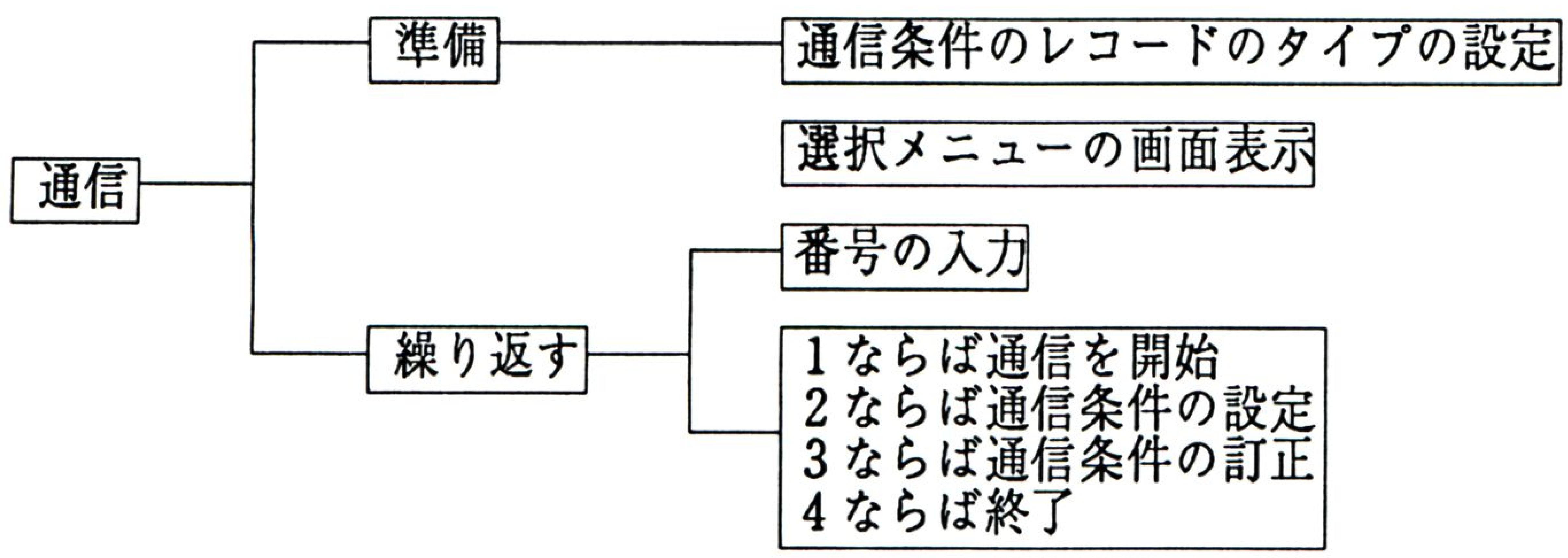
（7）バックスペースのサブプロシージャ

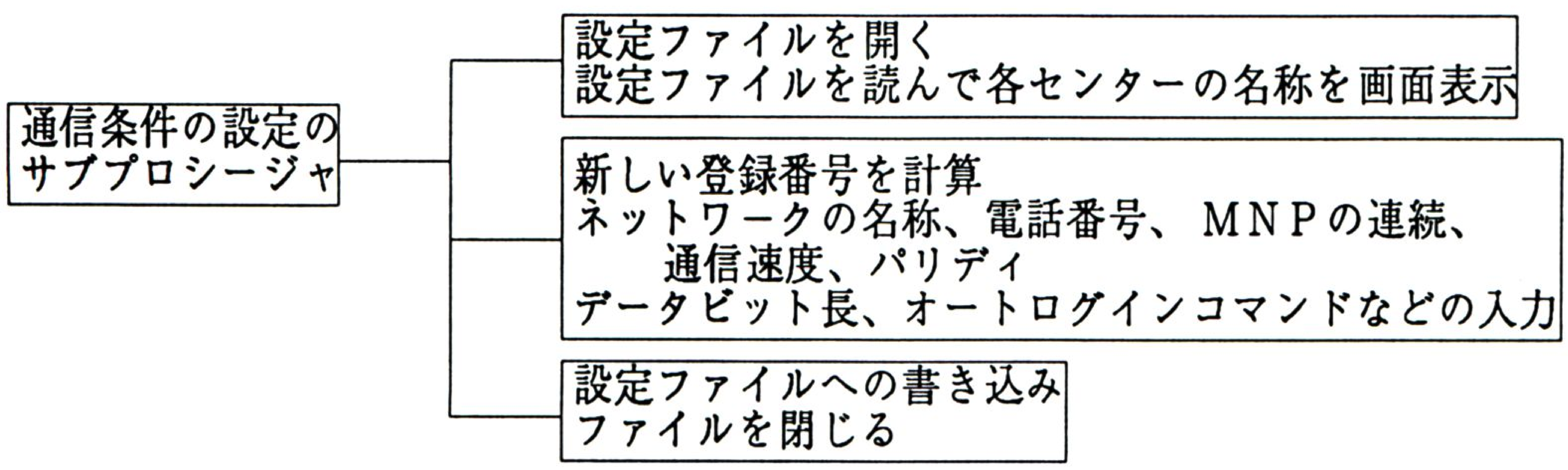
```
SUB BackSpace                   '←バックスペースのサブプロシージャ
    (省略　前節に同じ)
END SUB
```

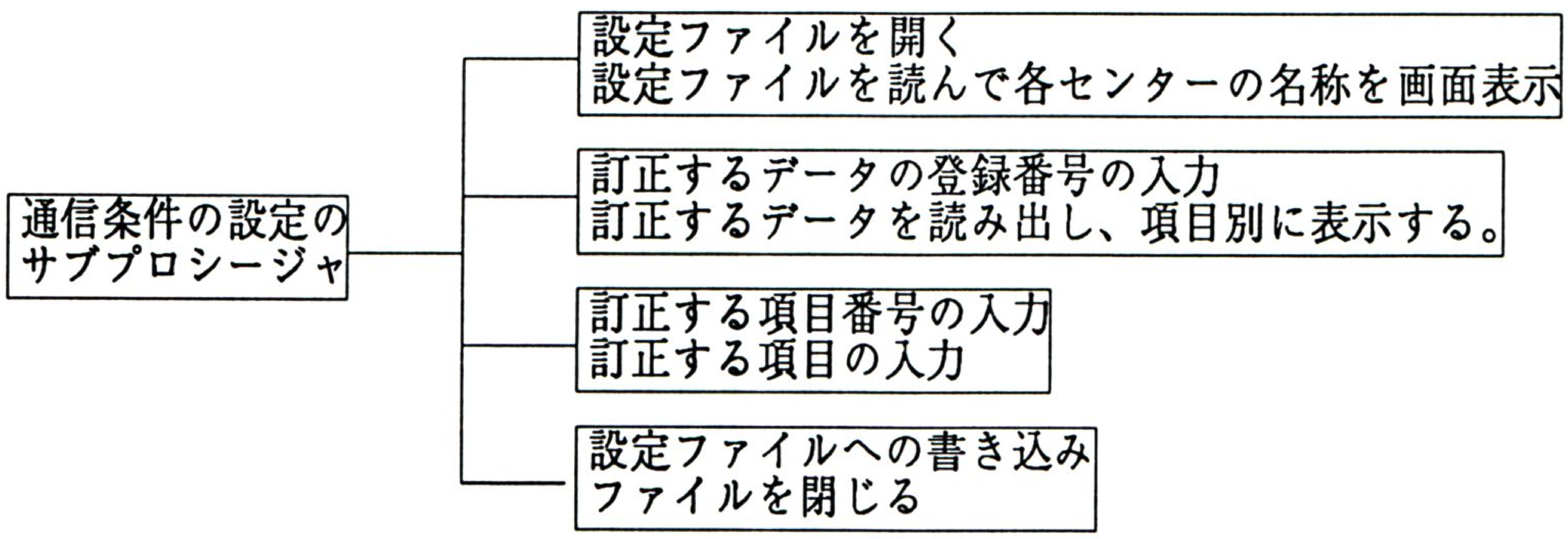
9．4　通信条件の設定

前節までのプログラムでは、パソコン通信のセンターごとに、別のプログラムを必要としました。そこで、あらかじめ、パソコン通信の通信条件をファイルに入力し、これを読み出して利用するように改造します。

プログラムの改造の概要は次のとおりです。







（1）メインプロシージャ

```
DEFINT A-Z
DECLARE SUB KeySet ()
DECLARE SUB CenterSentaku ()
DECLARE SUB Setuzoku (KakuninData$)
DECLARE SUB SouZyusin ()
```

```
DECLARE SUB Settei ()
DECLARE SUB Teisei ()
DECLARE SUB Zyusin (M$)
DECLARE SUB BackSpace ()
DECLARE SUB Sousin (SousinData$)
DECLARE SUB SpaceCut (SousinData$, S$)
DECLARE SUB AtSousin (SousinData$, KakuninData$)
'通信4    ****  通信条件の設定と訂正  ****(TUUSIN4.BAS)
TYPE SetteiType                           '←オートログインのレコードタイプ
    Namae  AS STRING * 20                      '←センターの名称
    Denwa  AS STRING * 16                      '←センターの電話番号
    Mnp    AS STRING * 1                       '←MNP接続
    Sokudo AS STRING * 4                       '←通信速度
    Parity AS STRING * 1                       '←パリティ
    Nagasa AS STRING * 1                       '←データ長
    StopBit AS STRING * 1                      '←ストップビット
    Setuzoku AS STRING * 20                    '←電話接続後の送信信号
    IdPrompt AS STRING * 10                    '←ＩＤ番号のプロンプト
    IdBangou AS STRING * 10                    '←ＩＤ番号
    PassPrompt AS STRING * 10                  '←パスワードのプロンプト
    PassWord  AS STRING * 10                   '←パスワード
END TYPE
DIM SetteiR AS SetteiType                      '←レコード
Hazime:                                        '←はじめの名札
DO
    CLS                                        '←画面消去
    PRINT "1: 通信の開始": PRINT
    PRINT "2: 通信条件の設定": PRINT
    PRINT "3: 通信条件の訂正": PRINT
    PRINT "4: 終了": PRINT
    PRINT "選択して番号を入力してください。-->";
    DO: K$ = INKEY$: LOOP UNTIL K$ >= "0" AND K$ <= "4"
    PRINT
    SELECT CASE K$
        CASE "1"                                   '←1ならば
            CenterSentaku                          '←センターを選択して
            Setuzoku KakuninData$                  '←接続する
            IF KakuninData$ = "CONNECT" THEN       '←CONNECTとかえってきたら
                SouZyusin                          '←送受信を開始
            END IF
        CASE "2": Settei                       '←2ならば、通信条件の設定
        CASE "3": Teisei                       '←3ならば、通信条件の訂正
        CASE "4": CLS : CLOSE : END            '←4ならば、終了
    END SELECT
LOOP
END
```

（２）通信条件の設定のサブプロシージャ

```
SUB Settei                                  '←接続条件等の設定
SHARED SetteiR AS SetteiType                     '←SetteiRの共用
CLS 2                                            '←文字画面消去
PRINT
PRINT "******** 電話番号等の設定 **********": PRINT
OPEN "B:Settei.dat" FOR RANDOM AS #2 LEN = LEN(SetteiR)
                                   '←通信条件を設定したデータのファイルを開く
FOR I = 1 TO LOF(2) / LEN(SetteiR)
    GET #2, I, SetteiR                           '←設定ファイルを読んで
    PRINT I; "   "; SetteiR.Namae                '←画面表示する
NEXT I
TourokuBangou = LOF(2) / LEN(SetteiR) + 1        '←登録番号の決定
PRINT : PRINT "登録番号-->"; TourokuBangou: PRINT
INPUT "通信ネットワークの名称                    -->", SetteiR.Namae
INPUT "電話番号（例  0123-45-6789）              -->", SetteiR.Denwa
INPUT "ＭＮＰ接続（しない:0  する:1 ）           -->", SetteiR.Mnp
INPUT "通信速度（例  1200、2400）                -->", SetteiR.Sokudo
INPUT "パリティ（例  なし:N  偶数:E  奇数:O）    -->", SetteiR.Parity
INPUT "データビット長（7ビット:7  8ビット:8）    -->", SetteiR.Nagasa
INPUT "ストップビット（1  あるいは  2）          -->", SetteiR.StopBit
PRINT "接続直後に送信する信号"
PRINT "（ﾘﾀｰﾝ:!  ﾌﾟﾛﾝﾌﾟﾄ:[]  約0.5秒待つ:* ）"
PRINT "  （一般のＢＢＳの場合  !）"
PRINT "  （ＰＣ－ＶＡＮの場合  VANPCNEC02!）"
PRINT "  （NIFTY Serveの場合  .01+[->]SVC!）"
INPUT "                                          -->", SetteiR.Setuzoku
INPUT "ＩＤ番号のプロンプト（[]で囲む  [->]）    -->", SetteiR.IdPrompt
INPUT "ＩＤ番号（例  abcd1234! ）                -->", SetteiR.IdBangou
INPUT "パスワードのプロンプト（[]で囲む[->]）    -->", SetteiR.PassPrompt
INPUT "パスワード（例  abcd1234! ）              -->", SetteiR.PassWord
PUT #2, TourokuBangou, SetteiR               '←設定のファイルへの書き込み
CLOSE #2                                     '←ファイルを閉じる
END SUB
```

（３）通信条件を訂正するサブプロシージャ

```
SUB Teisei                               '←訂正のサブプロシージャ
SHARED SetteiR AS SetteiType                 '←SetteiRの共用
CLS 2
    PRINT
    PRINT "******** 電話番号等の訂正 **********": PRINT
OPEN "B:Settei.dat" FOR RANDOM AS #2 LEN = LEN(SetteiR)
                                   '←通信条件の設定したデータのファイルを開く
FOR I = 1 TO LOF(2) / LEN(SetteiR)
    GET #2, I, SetteiR                           '←設定ファイルを読んで
```

```
    PRINT I; "   "; SetteiR.Namae              '←画面表示する
NEXT I
PRINT
PRINT "訂正するデータの登録番号を入力してください。-->";
DO: T$ = INKEY$: LOOP UNTIL T$ <> ""
TourokuBangou = VAL(T$)                        '←登録番号の入力
IF TourokuBangou > LOF(2) / LEN(SetteiR) OR TourokuBangou < 1 THEN
    CLOSE 2: EXIT SUB
END IF
GET #2, TourokuBangou, SetteiR         '←ファイルの登録番号のデータを読む
DO
  PRINT
  PRINT " 1  通信ネットワークの名称                    -->"; SetteiR.Namae
  PRINT " 2  電話番号                                  -->"; SetteiR.Denwa
  PRINT " 3  MNP接続（しない:0  する:1 ）              -->"; SetteiR.Mnp
  PRINT " 4  通信速度（300  1200  2400）               -->"; SetteiR.Sokudo
  PRINT " 5  パリティ（なし:N  偶数:E  奇数:O）        -->"; SetteiR.Parity
  PRINT " 6  データビット長（7ビット:7  8ビット:8）    -->"; SetteiR.Nagasa
  PRINT " 7  ストップビット（1  あるいは  2）          -->"; SetteiR.StopBit
  PRINT " 8  接続直後に送信する信号                    -->"; SetteiR.Setuzoku
  PRINT " 9  ID番号のプロンプト（[]で囲む  [->]）     -->"; SetteiR.IdPrompt
  PRINT "10  ID番号（例  abcd1234! ）                  -->"; SetteiR.IdBangou
  PRINT "11  パスワードのプロンプト（[]で囲む[->]）    -->"; SetteiR.PassPrompt
  PRINT "12  パスワード（例  abcd1234! ）              -->"; SetteiR.PassWord
  PRINT
  INPUT "訂正する項目の番号を入力してください。終了:E    -->";  TeiseiBangou$
  PRINT
  SELECT CASE VAL(TeiseiBangou$)
    CASE 1
      INPUT "通信ネットワークの名称                    -->", SetteiR.Namae
    CASE 2
      INPUT "電話番号（例  0123-45-6789 ）             -->", SetteiR.Denwa
    CASE 3
      INPUT "MNP接続（しない:0  する:1 ）              -->", SetteiR.Mnp
    CASE 4
      INPUT "通信速度（300  1200  2400）               -->", SetteiR.Sokudo
    CASE 5
      INPUT "パリティ（なし:N  偶数:E  奇数:O）        -->", SetteiR.Parity
    CASE 6
      INPUT "データビット長（7ビット:7  8ビット:8）    -->", SetteiR.Nagasa
    CASE 7
      INPUT "ストップビット（1  あるいは  2）          -->",SetteiR.StopBit
    CASE 8
      PRINT "接続直後に送信する信号"
      PRINT "   （一般のBBSの場合  リターン:!）"
      PRINT "   （PC－VANの場合  VANPCNEC!）"
      PRINT "   （NIFTY Serveの場合  .01+[->]SVC!）"
      INPUT "                                          -->",  SetteiR.Setuzoku
    CASE 9
```

```
        INPUT "ＩD番号のプロンプト（[]で囲む  [->]）-->",  SetteiR.IdPrompt
    CASE 10
        INPUT "ＩD番号（例  abcd1234!）              -->",  SetteiR.IdBangou
    CASE 11
        INPUT "パスワードのプロンプト（[]で囲む[->]）-->",  SetteiR.PassPrompt
    CASE 12
        INPUT "パスワード（例  abcd1234!）           -->",  SetteiR.PassWord
  END SELECT
LOOP UNTIL TeiseiBangou$ = "E"       '←TeiseiBangou$がEならば次へ
PUT #2, TourokuBangou, SetteiR       '←設定のデータをファイルへ書き込む
CLOSE #2                             '←ファイルを閉じる
END SUB
```

（4）通信先を選択するサブプロシージャ

```
SUB CenterSentaku                 '←センターを選択するサブプロシージャ
SHARED SetteiR AS SetteiType              '←SetteiRの共用
CLS 2                                     '←文字画面の消去
OPEN "B:Settei.dat" FOR RANDOM AS #2 LEN = LEN(SetteiR)
                            '←通信条件を設定したデータのファイルを開く
FOR I = 1 TO LOF(2) / LEN(SetteiR)
    GET #2, I, SetteiR                    '←設定ファイルを読んで
    PRINT I; "  "; SetteiR.Namae          '←画面表示する
NEXT I
PRINT : PRINT "どちらへ通信しますか。番号を入力してください。-->";
DO
    I$ = INKEY$                      '←通信するセンターの番号を入力
LOOP UNTIL VAL(I$) > 0 AND VAL(I$) <= LOF(2) / LEN(SetteiR)
GET #2, VAL(I$), SetteiR             '←設定のファイルを読む
VIEW PRINT 1 TO 1
LOCATE 1, 1
PRINT "     通信先   "; SetteiR.Namae; "  （"; SetteiR.Denwa; "）";
CLOSE #2                             'ファイルを閉じる
END SUB
```

（5）通信条件のデータのスペースを削除するサブプロシージャ

```
SUB SpaceCut (SousinData$, S$)
                        '←送信データの後部のスペースをカットする
S$ = ""
FOR I = LEN(SousinData$) TO 1 STEP -1
    IF MID$(SousinData$, I, 1) = " " THEN     '←後部がスペースならば
        SousinData$ = MID$(SousinData$, 1, I - 1)
                                              '←スペースをカットする
    ELSE
        S$ = SousinData$                      '←スペースがなければ
```

```
        EXIT SUB                          '←サブプロシージャを出る
    END IF
NEXT I
END SUB
```

（6）接続のサブプロシージャ

```
SUB Setuzoku (KakuninData$)              '←接続のサブプロシージャ
SHARED SetteiR AS SetteiType                  '←SetteiRの共用
VIEW PRINT 2 TO 22
CLS 2                                         '←文字画面消去
LOCATE 2, 1, 1, 14, 15                        '←カーソルを表示する
FILEMEI$ = "COM1:" + SetteiR.Sokudo + "," + SetteiR.Parity + ",
        " + SetteiR.Nagasa + "," + SetteiR.StopBit + ",CD0,DS0"
OPEN FILEMEI$ FOR RANDOM AS #1 LEN = 256       '←通信回線のファイルをく
Kaigyou$ = CHR$(&HD)                           '←改行
LineFeed$ = CHR$(&HA)                          '←ラインフィード
OK$ = "OK" + Kaigyou$ + LineFeed$              '←OK$の設定
AtSousin "ATZ", OK$                            '←モデムの初期化
AtSousin "ATP1", OK$                           '←自分の電話の方式
AtSousin "ATB0", OK$                           '←CCITT規格
AtSousin "ATX4", OK$                           '←接続後の通信速度の表示
SELECT CASE SetteiR.Mnp
    CASE "0": AtSousin "AT¥V0", OK$            '←MNP接続しない
              AtSousin "AT¥N0", OK$
    CASE "1": AtSousin "AT¥V2", OK$            '←MNP接続する
              AtSousin "AT¥N3", OK$
END SELECT
KakuninData$ = "CONNECT"
AtSousin "ATD" + SetteiR.Denwa, KakuninData$        '←ダイヤル
IF KakuninData$ = "BUSY" THEN                       '←話中のときは
    CLOSE                                          '←ファイルを閉じて
    PRINT "  話中です。 何かキーを押してください。"
    DO: LOOP UNTIL INKEY$ <> ""                    '←何かキーを押したら
    EXIT SUB                                       '←はじめに戻る
END IF
AtSousin "", Kaigyou$ + LineFeed$             '←改行とラインフィードの確認
Sousin SetteiR.Setuzoku                       '←接続直後の信号
Sousin SetteiR.IdPrompt                       '←ID番号のプロンプト
Sousin SetteiR.IdBangou                       '←ID番号
Sousin SetteiR.PassPrompt                     '←パスワードのプロンプト
Sousin SetteiR.PassWord                       '←パスワード
KakuninData$ = "CONNECT"                      '←確認のためにCONNECTのセット
END SUB
```

（7）ＡＴコマンドを送信するサブプロシージャ

```
SUB AtSousin (SousinData$, KakuninData$)    '←送信のサブプロシージャ
IF SousinData$ <> "" THEN                   '←送信データがあれば
    (省略　前節に同じ)
END SUB                                     '←サブプロシージャを出る
```

（8）オートログインコマンドを送信するサブプロシージャ

```
SUB Sousin (SousinData$)
                '←自動接続のためのコマンドを送信するサブプロシージャ
  (省略　前節に同じ)
END SUB
```

（9）送受信するサブプロシージャ

```
SUB SouZyusin                   '←送受信のサブプロシージャ
  (省略　前節に同じ)
END SUB
```

（１０）受信のサブプロシージャ

```
SUB Zyusin (M$)                 '←受信のサブプロシージャ
  (省略　前節に同じ)
END SUB
```

（１１）バックスペースのサブプロシージャ

```
SUB BackSpace               '←バックスペースのサブプロシージャ
  (省略　前節に同じ)
END SUB
```

9．5　アップロードとダウンロード

パソコン通信で多量の情報を受信しても、これを画面上で読み取ることは困難です。そこで、送受信した記録をファイルとして保存する必要があります。このことをダウンロードといいます。

また、逆に、電子掲示板や電子メールへ送信する場合に、接続した状態でキーボードから入力していたのでは、接続時間が長くなり、電話料や接続料金の負担が多額になる恐れもあります。そこで、接続する前に、あらかじめ、送信内容を入力してファイルとしておき、接続後、一括してファイルの内容を送信すれば、短時間で済み、経費の節約にもなります。このことをアップロードといいます。

アップロードとダウンロードは、接続後、通信の途中で、ファンクションキーを押して利用します。従って、プログラム上では途中で割り込む方法で作動させることになります。

およそ、次のような手順でプログラムをつくります。

（1）ファンクションキーに機能を表示します。

アップーロードをファンクションキー6番へ、ダウンロードをファンクションキー7番へ割り当てます。

KEY 6, "記録始"

KEY 7, "Fi送信"　　　Fiは、ファイルを送信するという意味で、FileのFiからとりました。

KEY 6, "記録終"　　　通信記録中の表示

（2）表示します。

KEY ON

（3）イベントトラッピングを設定します。

ON KEY(6) GOSUB KirokuHazime

ON KEY(7) GOSUB FileSousin

ON KEY(6) GOSUB KirokuOwari　　　通信記録を保存しているときは、ファンクション6を終了に割り当てます。

（4）機能をＯＮにします。

KEY(6) ON

KEY(7) ON

（7）KirokuHazime、KirokuOwari及びFileSousinのサブルーチンをメインプログラムに加えます。

サブルーチンのプログラムの構造は次のとおりです。

通信記録の保存開始のサブルーチン	カーソルの現在位置を取得 記録を保存するシルシを付ける 保存するファイル名を入力する 保存するファイルを開く 通信記録をファイルへ保存していることを表示する ファンクションキー6に記録終了を割り当てて表示する 元の位置へカーソルを戻す
通信記録の保存終了のサブルーチン	カーソルの現在位置を取得 通信記録のファイルを閉じる ファンクションキー6に通信記録開始を割り当てて表示する 元の位置へカーソルを戻す
ファイルを一括送信するサブルーチン	カーソルの現在位置を取得 送信するファイル名を入力する 送信するファイルを開く ファイルへ一括送信していることを表示する ファイルが終りになるまでファイルを読んで送信することを繰り返す 元の位置へカーソルを戻す

（1）メインプロシージャ

```
DEFINT A-Z
DECLARE SUB KeySet ()
DECLARE SUB CenterSentaku ()
DECLARE SUB Setuzoku (KakuninData$)
DECLARE SUB SouZyusin ()
DECLARE SUB Settei  ()
DECLARE SUB Teisei  ()
DECLARE SUB Zyusin (M$)
DECLARE SUB BackSpace ()
DECLARE SUB Sousin (SousinData$)
DECLARE SUB SpaceCut (SousinData$, S$)
DECLARE SUB AtSousin (SousinData$, KakuninData$)
'通信５    ****  アップロードとダウンロード  ****  (TUUSIN5.BAS)
                                       '（通信記録の保存とファイルの一括送信）
TYPE SetteiType                         '←オートログインのレコードタイプ
    Namae   AS STRING * 20                 '←センターの名称
    Denwa   AS STRING * 16                 '←センターの電話番号
    Mnp     AS STRING * 1                  '←ＭＮＰ接続
    Sokudo  AS STRING * 4                  '←通信速度
    Parity  AS STRING * 1                  '←パリティ
    Nagasa  AS STRING * 1                  '←データ長
    StopBit AS STRING * 1                  '←ストップビット
    Setuzoku AS STRING * 20                '←電話接続後の送信信号
    IdPrompt AS STRING * 10                '←ＩＤ番号のプロンプト
    IdBangou AS STRING * 10                '←ＩＤ番号
    PassPrompt AS STRING * 10              '←パスワードのプロンプト
```

```
    PassWord  AS STRING * 10                    '←パスワード
END TYPE
DIM SetteiR AS SetteiType                       '←レコード
Hazime:                                         '←はじめの名札
DO
    CLS                                         '←画面消去
    KeySet                                    '←ファンクションキーのセット
    PRINT "1: 通信の開始": PRINT
    PRINT "2: 通信条件の設定": PRINT
    PRINT "3: 通信条件の訂正": PRINT
    PRINT "4: 終了": PRINT
    PRINT "選択して番号を入力してください。-->";
    DO: K$ = INKEY$: LOOP UNTIL K$ >= "0" AND K$ <= "4"
    PRINT
    SELECT CASE K$
        CASE "1"                                '←1ならば
            CenterSentaku                       '←センターを選択して
            Setuzoku KakuninData$               '←接続する
            IF KakuninData$ = "CONNECT" THEN    '←CONNECTとかえってきたら
                SouZyusin                       '←送受信を開始
            END IF
        CASE "2": Settei                        '←2ならば、通信条件の設定
        CASE "3": Teisei                        '←3ならば、通信条件の訂正
        CASE "4": CLS : CLOSE : END             '←4ならば、終了
    END SELECT
LOOP
END
```

（２）通信記録の保存開始（ダウンロード開始）のサブルーチン

```
KirokuHazime:                                '←ファイルを保存するサブルーチン
  y = CSRLIN: X = POS(0)                        '←カーソルの現在位置を取得
  Kiroku = 1                                    '←記録を保存するサイン
  VIEW PRINT 23 TO 23
  CLS 2                                         '←文字画面消去
  PRINT "保存するファイル名を入力してください。（例  B:HOZON.LOG）-->";
  INPUT HozonFile$                              '←保存するファイル名の入力
  OPEN HozonFile$ FOR OUTPUT AS #3              '←保存するファイル名を開く
  CLS 2                                         '←文字画面消去
  PRINT "******* 通信記録を "; HozonFile$; " へ保存しています。 ********";
  KEY 6, "記録終"
  KEY ON
  KEY(6) OFF
  ON KEY(6) GOSUB KirokuOwari
  KEY(6) ON
  VIEW PRINT 2 TO 22
  LOCATE y, X                                   '←カーソルを戻す
RETURN
```

（３）通信記録の保存終了（ダウンロード終了）のサブルーチン

```
KirokuOwari:   '←通信記録の保存を終了してファイルを閉じるサブルーチン
    y = CSRLIN: X = POS(0)              '←カーソルの現在位置を取得
    CLOSE 3
    Kiroku = 0                          '←記録を保存するサイン
    VIEW PRINT 23 TO 23
    CLS 2                               '←文字画面消去
    KEY 6, "記録始": KEY ON: KEY(6) OFF
    ON KEY(6) GOSUB KirokuHazime
    KEY(6) ON
    VIEW PRINT 2 TO 22
    LOCATE y, X                         '←カーソルを戻す
RETURN
```

（４）ファイルの一括送信（アップロード）のサブルーチン

```
FileSousin:                        '←ファイルを一括送信するサブルーチン
  y = CSRLIN: X = POS(0)           '←カーソルの現在位置を取得
  VIEW PRINT 24 TO 24
  CLS 2                            '←文字画面消去
  PRINT "送信するファイル名を入力してください。（例  B:SOUSIN.FIL ）-->";
  INPUT SousinFile$                '←送信するファイル名を入力
  OPEN SousinFile$ FOR INPUT AS #4 '←送信するファイル名を開く
  CLS 2                            '←文字画面消去
  PRINT "******* ファイル  "; HozonFile$; " を一括して送信しています。 ********"
  VIEW PRINT 2 TO 22
  DO
     LINE INPUT #4, SousinRec$     '←ファイルを読む
     PRINT #1, SousinRec$          '←ファイルを送信する
  LOOP UNTIL EOF(4)                '←ファイルが終わりになるまで
  CLOSE 4                          '←ファイルを閉じる
  VIEW PRINT 24 TO 24
  CLS 2                            '←文字画面消去
  VIEW PRINT 2 TO 22
 LOCATE y, X                       '←カーソルを戻す
RETURN
```

（５）ファンクションキーを設定するサブプロシージャ

```
SUB KeySet           '←ファンクションキーを設定するサブプロシージャ
KEY 6, "記録始"      '←ファンクションキー6は受信記録のファイル保存
KEY 7, "Fi送信"      '←ファンクションキー7はファイルの一括送信
KEY ON                         '←ファンクションキーの画面表示
ON KEY(6) GOSUB KirokuHazime   '←f･6を押すとファイル保存のサブプロシージャへ
ON KEY(7) GOSUB FileSousin     '←f･7を押すとファイル送信のサブプロシージャへ
KEY(6) ON                      '←f･6の機能をON
KEY(7) ON                      '←f･7の機能をON
LINE (0, 0)-(639, 16), 1, BF
```

```
LINE (0, 353)-(639, 368), 1, BF
LINE (0, 369)-(639, 384), 5, BF
LINE (0, 385)-(639, 399), 4, BF
VIEW PRINT 2 TO 22              '←ファンクションの表示が消えないように
END SUB
```

（6）受信のサブプロシージャ

```
SUB Zyusin (M$)                        '←受信のサブプロシージャ
SHARED Kiroku
    M$ = ""
    IF LOC(1) THEN                     '←入力待ちの文字があるならば
        M$ = INPUT$(1, #1)             '←1バイトを読む
        SELECT CASE M$
            CASE IS >= CHR$(&H20)      '←コードが16進数20以上の場合は
                IF (M$ >= CHR$(&H80) AND M$ <= CHR$(&H9F))
                    OR (M$ >= CHR$(&HE0) AND M$ <= CHR$(&HEF)) THEN
                    Kanji$ = M$
                        '↑漢字コードの1バイト目ならば←1バイト目を保存し
                    M$ = INPUT$(1, #1)      '←2バイト目を読み取って
                    M$ = Kanji$ + M$        '←1バイト目と2バイト目を連結する
                END IF
                PRINT M$;                   '←データを画面表示する
            CASE CHR$(13)                   '←コードが改行ならば
                PRINT                       '←画面上で改行する
            CASE CHR$(8)                    '←バックスペースならば
                BackSpace                   '←バックスペースの手続き
        END SELECT
        IF Kiroku THEN                      '←記録保存なら
            PRINT #3, M$;                   '←ファイルへ書き込む
        END IF
    END IF
END SUB
```

（以下のサブプロシージャは前節と同じなので省略します。）

```
SUB AtSousin (SousinData$, KakuninData$)  '←送信のサブプロシージャ
SUB BackSpace                  '←バックスペースのサブプロシージャ
SUB CenterSentaku              '←センターを選択するサブプロシージャ
SUB Settei                         '←接続条件等の設定
SUB Setuzoku (KakuninData$)        '←接続のサブプロシージャ
SUB Sousin (SousinData$)
                   '←自動接続のためのコマンドを送信するサブプロシージャ
SUB SouZyusin                      '←送受信のサブプロシージャ
SUB SpaceCut (SousinData$, S$)
                          '←送信データの後部のスペースをカットする
SUB Teisei                '←訂正のサブプロシージャ
```

９．６　通信ソフトのまとめ

最後に、電話を切る機能、プログラムの先頭に戻す機能及びセンター側にブレーク信号（処理の中断を求める信号）を送信する機能を加えて、まとめとします。

電話を切る機能及びプログラムの先頭に戻す機能のプログラムは次のようにつくります。

（１）ファンクションキー２に電話を切る機能を割り当て、ファンクションキー３にプログラムの先頭に戻る機能を割り当てます。

```
KEY 2, "電話切"
KEY 3, "最初へ"
```

（２）画面表示します。

（３）機能を設定します。

```
ON KEY(2) GOSUB DenwaSetudan
ON KEY(3) GOSUB Modoru
```

（４）機能をＯＮにします。

```
KEY(2) ON
KEY(3) ON
```

（５）サブルーチンプログラムをつくります。

① モデムが１秒間以上、何もしない時間をとります。

② モデムへ+++を送ります。（改行しません）　　PRINT #1,"+++";

③ OKが返ってきたら、ATHを送ります。

④ ファイルを閉じます。

⑤ Hazimeへとばします。

１秒以上停止した後に+++を送ると、オンラインモードからコマンドモードへ切り替わります。そこで、ATHという電話を切るコマンドを送ります。

GOTO Hazime は、あまり使わないほうがよいのですが、ここでは緊急避難的に使いました。

プログラムの先頭には、Hazime:という名札を付けておいて、その位置へプログラムが進むようにしました。

ブレーク信号の機能は、センター側の処理の中断を求めるものです。例えば、掲示板を読む場合に、長いメッセージの受信を中断して次に進みたい場合に使います。しかし、実は、ＭＮＰを使用しているときは、効き目がよくないようです。なぜなら、ＭＮＰはモデム間のパケット通信のようなものですし、長いブレーク信号を送っても、機構上３００ミリ秒のブレーク信号になって送られるからです。

ブレーク信号のプログラムは、ＩＯポートにコマンドを送ることによって、ブレー

ク信号をＲＳ２３２Ｃからモデムへ送信します。手順は次のとおりです。

（１）ブレーク信号ＯＮを送ります。

　ＩＯポートアドレス＆Ｈ３２へ、コマンド＆Ｈ３Ｆを送ります。

　OUT &H32, &H3F

（２）時間をおきます。

　ＴＩＭＥＲから秒数を読み取って、１秒以上の時間をおきます。

（３）ブレーク信号ＯＦＦを送ります。

　ＩＯポートアドレス＆Ｈ３２へ、コマンド＆Ｈ３７を送ります。

　OUT &H32, &H37

（１）メインプロシージャ

```
DEFINT A-Z
 （省略）
DECLARE SUB AtSousin (SousinData$, KakuninData$)
'通信                                    (TUUSIN6.BAS)
TYPE SetteiType                              '←オートログインのレコードタイプ
    Namae    AS STRING * 20                     '←センターの名称
    Denwa    AS STRING * 16                     '←センターの電話番号
    Mnp      AS STRING * 1                      '←ＭＮＰ接続
    Sokudo   AS STRING * 4                      '←通信速度
    Parity   AS STRING * 1                      '←パリティ
    Nagasa   AS STRING * 1                      '←データ長
    StopBit AS STRING * 1                       '←ストップビット
    Setuzoku AS STRING * 20                     '←電話接続後の送信信号
    IdPrompt AS STRING * 10                     '←ＩＤ番号のプロンプト
    IdBangou AS STRING * 10                     '←ＩＤ番号
    PassPrompt AS STRING * 10                   '←パスワードのプロンプト
    PassWord    AS STRING * 10                  '←パスワード
END TYPE
DIM SetteiR AS SetteiType                       '←レコード
 Hazime:                                        '←はじめの名札
DO
    VIEW PRINT 1 TO 25
    CLS                                         '←画面消去
    KeySet                                    '←ファンクションキーのセット
    Kiroku = 0
    PRINT "1: 通信の開始": PRINT
    PRINT "2: 通信条件の設定": PRINT
    PRINT "3: 通信条件の訂正": PRINT
    PRINT "4: 終了": PRINT
    PRINT "選択して番号を入力してください。-->";
    DO: K$ = INKEY$: LOOP UNTIL K$ >= "0" AND K$ <= "4"
    PRINT
    SELECT CASE K$
        CASE "1"                                 '←1ならば
```

```
            CenterSentaku                       '←センターを選択して
            Setuzoku KakuninData$               '←接続する
            IF KakuninData$ = "CONNECT" THEN
                                                '←CONNECTとかえってきたら
                SouZyusin                       '←送受信を開始
            END IF
        CASE "2": Settei                        '←2ならば、通信条件の設定
        CASE "3": Teisei                        '←3ならば、通信条件の訂正
        CASE "4"
            VIEW PRINT 1 TO 25: CLS : CLOSE : END  '←4ならば、終了
    END SELECT
LOOP
END
```

（2）電話を切断するサブルーチン

```
DenwaSetudan:                                   '←電話を切断するサブルーチン
    T& = TIMER                                  '←タイマーから秒数を複写
    DO: LOOP UNTIL TIMER > (T& + 1)             '←1秒以上を経過したら次へ
    PRINT #1, "+++";                            '←モデムをコマンドモードへ切り替え
    T& = TIMER                                  '←タイマーから秒数を複写
    DO: LOOP UNTIL TIMER > (T& + 1)             '←1秒以上を経過したら次へ
    AtSousin "", OK$
    AtSousin "ATH", "OK"                        '←モデム停止、電話切断
Modoru:                                         '←メインへ戻るサブルーチン
    CLOSE                                       '←ファイルを閉じる
    GOTO Hazime                                 '←メインへ戻る
RETURN
```

（3）ブレーク信号を送るサブルーチン

```
BreakSingou:                                    '←ブレーク信号を送るサブルーチン
    OUT &H32, &H3F                              '←ブレーク信号ONを送る
    T& = TIMER
    DO: LOOP UNTIL TIMER > T& + 1               '←時間をおく
    OUT &H32, &H37                              '←ブレーク信号OFFを送る
RETURN
```

```
KirokuHazime:                                   '←ファイルを保存するサブルーチン
（省略）
RETURN
```

```
KirokuOwari:        '←通信記録の保存を終了してファイルを閉じるサブルーチン
（省略）
RETURN
```

```
FileSousin:                    '←ファイルを一括送信するサブルーチン
（省略）
RETURN
```

（4）ファンクションキーを設定するサブプロシージャ

```
SUB KeySet              '←ファンクションキーを設定するサブプロシージャ
KEY 2, "電話切"            '←ファンクションキー2は電話切る
KEY 3, "最初へ"            '←ファンクションキー3はメインの最初に戻る
KEY 6, "記録始"            '←ファンクションキー6は受信記録のファイル保存
KEY 7, "Fi送信"            '←ファンクションキー7はファイルの一括送信
KEY 9, ":BREAK"           '←ファンクションキー9はブレーク信号の送信
KEY ON                    '←ファイクションキーの画面表示
ON KEY(2) GOSUB DenwaSetudan   '←f·1を押すと電話切断のサブプロシージャへ
ON KEY(3) GOSUB Modoru         '←f·2を押すとメインへ戻るサブプロシージャへ
ON KEY(6) GOSUB KirokuHazime   '←f·6を押すとファイル保存のサブプロシージャへ
ON KEY(7) GOSUB FileSousin     '←f·7を押すとファイル送信のサブプロシージャへ
ON KEY(9) GOSUB BreakSingou   '←f·9を押すとブレイク信号送信のサブプロシージャへ
KEY(2) ON                          '←f·2の機能をON
KEY(3) ON                          '←f·3の機能をON
KEY(6) ON                          '←f·6の機能をON
KEY(7) ON                          '←f·7の機能をON
KEY(9) ON                          '←f·9の機能をON
LINE (0, 0)-(639, 16), 1, BF
LINE (0, 353)-(639, 368), 1, BF
LINE (0, 369)-(639, 384), 5, BF
LINE (0, 385)-(639, 399), 4, BF
VIEW PRINT 2 TO 22             '←ファンクションの表示が消えないように
END SUB
```

（以下のサブプロシージャは前節に同じなので省略します。）

```
SUB AtSousin (SousinData$, KakuninData$)   '←送信のサブプロシージャ
SUB BackSpace                  '←バックスペースのサブプロシージャ
SUB CenterSentaku              '←センターを選択するサブプロシージャ
SUB Settei                     '←接続条件等の設定
SUB Setuzoku (KakuninData$)    '←接続のサブプロシージャ
SUB Sousin (SousinData$)
                     '←自動接続のためのコマンドを送信するサブプロシージャ
SUB SouZyusin                 '←送受信のサブプロシージャ
SUB SpaceCut (SousinData$, S$)
                              '←送信データの後部のスペースをカットする
SUB Teisei                    '←訂正のサブプロシージャ
SUB Zyusin (M$)               '←受信のサブプロシージャ
```

INDEX

ステートメント関数	ページ	機　能
GOSUB	104	サブルーチンにジャンプする
'$INCLUDE	147,218	プロシージャ定義モジュール使用宣言
IF __ THEN __ ELSE	37,85,88,89	論理式の条件判断を行う
INKEY$関数	70	キー入力した文字を読み込む
INPUT	36	データを入力する
INPUT$関数	258	指定したバイト数の文字列を読み込む
INPUT¥関数	252,253	指定した文字数の文字列を読み込む
INSTR関数	139,252	文字列中の指定文字列のバイト位置
KEY	271	ファンクションキーに文字列を定義する
KEY ON/OFF	196,271	ファンクションキーによる割込みの制御
KINSTR関数	139	文字列中の指定文字列のバイト位置
KMID$関数	83	指定した文字列の一部を取り出す
LEN 節	109,112	バイト長を得る
LINE	21,28,54	グラフィック画面上にラインをひく
LOC関数	252,253	ファイル内の位置を得る
LOCATE	101	テキスト画面上の行と桁の指示をする
LOF関数	136	ファイルの大きさを得る
LPRINT	39	プリンタにデータを出力する
MID$関数	82	指定した文字列の一部を取り出す
MOD演算子	59	剰余を返す
MouseBouder/MOUSE.BI	228	マウスの移動範囲の設定
MouseHide / 〃	230,236	マウスの非表示
MouseInit / 〃	218,219	マウスの初期化
MouseLocate/ 〃	228	マウスの表示位置の設定
MouseMode / 〃	223	マウスのタイプの設定
MousePoll / 〃	219	マウスの現在の位置等の取得
MousePush / 〃	218,219	ボタンを押した時の位置等の取得
MouseReady%/ 〃	230	マウスが利用可能の判定関数
MouseRelease/ 〃	230	ボタンを離した時の位置等の取得
MouseShow / 〃	218,219	マウスカーソルの表示
SetMouseEvent/ 〃	230,236	マウス割込み条件の設定
ON KEY() GOSUB	195,196,271	ファンクションキーによる割込み
ON PLAY() GOSUB	104	ＢＧＭデータバッファ状態での割込み
ON UEVENT GOSUB	235,237	ユーザー定義イベントの割込み
OPEN "COMn:"	251	通信ポートのファイルを開く

ステートメント関数	ページ	機　能
OPEN _ FOR OUTPUT	129	出力用シーケンシャルファイルを開く
OPEN _ FOR RANDOM	109,112	ランダムファイルを開く
OUT	278	Ｉ／Ｏポートに１バイト送出
PAINT	46	指定した領域を指定した色で塗りつぶす
PALETTE	101	パレット番号に色コードをセットする
PLAY	103	音楽演奏を行う
PLAY ON/OFF	104	ＢＧＭバッファの割込みを制御
POS(0)	196,197	テキスト画面のカーソル桁位置を得る
PRESET	59	グラフィック画面にドットを描く
PRINT	39	画面にデータを表示する
PRINT USING	62	指定書式で画面にデータを表示する
PSET	59	グラフィック画面にドットを描く
PUT	85,86	グラフィックパターンを表示する
PUT #	133,135	ファイルにデータを書き込む
RANDMIZE	43	乱数系列を初期化する
READ	80	DATAステートメントから値を読込み変数に代入
RETURN	105	サブルーチンから戻る
RND関数	42	０～１の範囲の単精度の乱数を返す
SCREEN関数	101	グラフィック画面のモードを設定する
SELECT CASE	71	条件式の１つを実行する
CASE 節	71	数値式または文字列式
CASE IS 節	73	範囲を表す数値式または文字列式
SIN関数	238,241	ラジアン単位で指定した角度のサイン値
SOUND	106	スピーカーから音を出す
SUB _ END SUB	74	サブプログラムの始めと終り
SWAP	121	２つの変数の値を入れ換える
TIMER関数	43	午前0時から現在までの秒数を返す
TYPE _ END TYPE	113	ユーザー定義のデータ型を宣言する
UEVENT ON/OFF	236,237,238	ユーザー定義イベントの割込みを制御
VAL	82,83	数値文字列を数値に変換する
VIEW	63	グラフィック画面の表示領域指定
VIEW PRINT	163	テキスト画面のスクロール範囲を指定
WINDOW	63	ビューポートの論理座標を指定
WRITE #	128	シーケンシャルファイルにデータ書込み

楽しく学べるQuick BASIC

1993年6月9日　第1刷発行

著者　松尾昭明

発行者　東京書籍株式会社
代表者　丁子　惇

印刷所　東京書籍印刷株式会社

発行所　東京書籍株式会社
東京都北区堀船2-17-1　〒114

ISBN-4-487-79085-9

東京書籍のコンピュータ関連商品

参考図書

ブラックボックス開けちゃった!! 堀口秀嗣監修 村松浩幸著

●コンピュータを学び始めた方々へ
B5判 124頁 定価2,600円(本体2,524円)

「計算する」「文字を書く」「図形を描く」「検索する」「通信する」…コンピュータの基本機能のしくみをサンプルプログラムに従って実習形成で体得できる。3.5/5インチフロッピー・ディスクつき。

パソコン・グラフィックス即戦力 井出 昭著

●授業に生きるパートナー
B5判 134頁 定価1,300円(本体1,262円)

シミュレーション・ソフト作成のための懇切丁寧な入門書。生徒使用もでき，実技講習のテキストとしても最適。NEC・富士通共通版。言語BASIC。

シミュレーション作成のための 実践テクニック集 井出 昭著

B5判 144頁 定価1,240円(本体1,204円)

主に中学校の数学の先生を対象として様々なテクニックの具体的かつ懇切な解説とサブルーチン用プログラム (FD「ユーティリティー」別売)の紹介。NEC98シリーズ用，言語BASIC。

パソコンソフト

情報基礎 コンピュータ・シミュレータ Gate Manager

定価19,400円(税別)
PC-9800シリーズ FMR-50シリーズ，FM-TOWNS対応
3.5インチまたは5インチプログラムデイスク1枚 学習ノート付

プログラミング学習の導入として命令の機能やコンピュータ内のデータの流れの概念をわかりやすく学習することを目的としたシミュレーション型のソフトウェア。

情報基礎用グラフィックワープロ「pair」

定価22,000円(税別)
PC-9800シリーズ(VX以降)

- 画面上で縦書きもでき，画面で見たままの形で印刷が可能。
- サンプルデータとして教科書の題材を取り上げ，指導案例を添付。

情報基礎用表計算&データベース「escal」

定価18,000円(税別)
PC-9800シリーズ(VX以降)

- 数表とグラフを同時に表現でき，グラフを描きながらデータ修正が可能。
- 豊富なサンプルデータを用意できる上，成績管理・図書管理等様々な用途にも充分活用できる。

TOWNS 情報基礎「SELECT-F」

定価30,000円(税別)
FM-TOWNS

- 情報基礎の学習内容に対応した日本語ワードプロセッサ・図形処理・表計算&グラフ作成・データベースの統合ソフト。
- 4機能を同一画面上で操作でき，データ表現のオーバーラップが可能。

漢字マッチ

定価7,800円(税別)
PC-9800シリーズ(VM以降)，FMR-50シリーズ，
FM-TOWNS対応

- 100種類の“へん”と“つくり”のカードから2枚を選んで，正しい漢字を作るゲーム感覚のソフト。

小学校算数 シミュレーション 4~6年

定価各9,700円(税別)
PC-9800シリーズ(VM以降)
FMR-50シリーズ，FM-TOWNS対応

- 一斉授業の形態をこわさず，授業の中に手軽にコンピュータを取り入れられる。
- 各ソフトの演示時間が数分以内に収まるようなショートプログラム構成。

中学校数学 ニューシミュレーション 図形編 1～3年

定価各9,900円(税別)
PC-9800シリーズ, FMR-50シリーズ, FM-TOWNS対応

教室にコンピュータが数台の環境でも使える演示形式のシミュレーションソフト。授業に容易に取り込める，ショートプログラム構成。だれでも使える親切な指導例付。

中学校数学 シミュレーション PartⅡ 1～3年

定価各7,800円(税別)
PC-9800シリーズ, FMR-50シリーズ, FM-TOWNS対応

- 数式・関数領域の演示形式のシミュレーションソフト。
- 授業に容易に取り込めるショートプログラム構成。
- だれでも使える親切な指導案例付。

中学校数学 課題学習用ソフト Let's try

定価各9,900円(税別)
PC-9800シリーズ, FMR-50シリーズ, FM-TOWNS対応

- 平成5年から実施される課題学習の教材のうち，黒板やプリントではイメージしにくい題材を選択。
- 22のテーマをショートプログラムにまとめ，授業に容易に取り組める。ワークシートと指導案例付。

LET'S PRACTICE 1～3年

定価各17,000円(税別)
PC-9800シリーズ(VM以降)

- 教科書の内容に対応した問題の中から必要な問題を呼びだして，レイアウト印刷することにより，プリント教材やテスト問題を簡単に作成できる。

中学校理科 理科グラフツール 実験GRAPH

定価9,690円(税別)
PC-9800シリーズ, FMR-50シリーズ, FM-TOWNS対応

実験名を選んだら，あとは測定値を入力するだけで即座にグラフ表示。実験誤差も一目でわかります。操作方法はすべて画面に表示されるマニュアルレス仕様です。指導案例付。

草花の観察「すみれ」

定価24,000円(税別)
PC-9800シリーズ, FMR-50シリーズ, FM-TOWNS対応

- 簡単な操作と美しいグラフィックスで身近な草花271種が検索できる。
- データは，市販のお絵かきソフトで簡単に追加・修正が可能。
- 授業だけでなく課外学習から生涯学習まで幅広く使える。

世界全図描画シミュレーション「エリシス」

定価19,400円(税別)
PC-9800シリーズ(VX以降)

- 33種類の図法を選択して，任意の地点を中心に正軸法・横軸法・斜軸法の世界地図が自由に描ける。
- 各種図形ソフトへデータ変換も可能。

栄養計算 HEALTHY Ⅱ

定価12,800円(税別)
PC-9800シリーズ, FMR-50シリーズ, FM-TOWNS対応

簡単な操作で食品成分表データを駆使して，楽しく献立作成ができるソフト「HEALTHY」を，さらに使いやすく，さらに機能アップした上で，10群と4群に，対応。

数学用図形作成&グラフ作成ソフト F-TOOL/SD

定価24,000円(税別)
PC-9800シリーズ(VX以降)

- 授業における電子黒板やプリント教材の作成に使える数学用図形作成&グラフ作成ソフト。
- 授業にすぐ役に立つ高校数Ⅰサンプルデータを添付。

高校理科 シミュレーションソフト

定価各36,000円(税別)
NEC PC-9800シリーズ対応

地学／地球の内部構造を調べる
生物／遺伝のしくみを調べる
化学／化学反応の速さ
物理／ボールの運動

- 個別学習（グループ学習）及び，先生の演示実験として用いることが可能。
- 生徒ワークシートとの連携による学習の展開。
- コンピュータ・グラフィックスによるビジュアルな画面。

＊このほかに英語教科書準拠ソフトがあります。
＊それぞれ追加フロッピー・学校パック21本等を用意してあります。

ISBN4-487-79085-9

C0055 P2800E

9784487790852

1910055028005

楽しく学べる

Quick BASIC

定価2,800円（本体2,718円）

東京書籍